MARCH 2024

MongoDB Atlas Security Controls

MongoDB®

Table of Contents

Table of Contents

Table of Contents

Introduction

Building customer trust is a top priority at MongoDB. We understand the responsibility we have when you, our customers, entrust us with a significant variety and amount of sensitive data. To maintain customer confidence in our security posture and in the security features we provide, we work diligently to continuously improve security processes and controls and provide our customers with the right features to secure the data. We take security seriously – from continuously fixing vulnerabilities and improving our security posture to enabling you to do just the same by providing various security features in our products. You will also find that we maintain and improve upon a full suite of security compliance certifications and attestations so as to keep up with the ever-changing threat and risk landscape.

At MongoDB, we want you to have full confidence in the security and resiliency of the systems and technology that we maintain, and the products that we provide to facilitate secure growth and innovation in your company. We are hopeful that this document conveys the depth of our commitment to customer trust by providing a detailed understanding of MongoDB Atlas security controls and features.

In addition to this document, we encourage you to review our Technical and Organizational Security Measures. The security measures set out the security features, processes, and controls applicable to the cloud services, including configurable options available to customers, which employ industry standard information security best practices.

What is MongoDB Atlas?

MongoDB Atlas is a fully managed cloud database with multi-cloud and multi-region data distribution capability. With automated infrastructure provisioning, database setup, maintenance, and version upgrades, customers can shift their focus to what really matters: building applications with speed and success. Atlas also comes with many drivers, tools, and a full suite of services (Atlas Search, Atlas Online Archive, Atlas Data Lake, and MongoDB App Services) to help our customers build to new heights securely.

MongoDB Atlas at a glance

Atlas provides a hierarchy based on organizations and projects to facilitate the management of your Atlas clusters. Multiple projects can exist within an organization. Billing happens at the organization level, though visibility into usage by the project is preserved.

By having multiple projects within an organization, you can:

- Isolate different environments from each other.
- Deploy into different regions or cloud platforms.
- Create different alert settings. For example, configure alerts for Production environments differently than Development environments.
- Maintain separate cluster security configurations. For example:
  - Create/manage different sets of MongoDB user credentials for each project.
  - Isolate networks in different VPCs.
- Associate different users or teams with different environments, or give different permissions to users in different environments.

MongoDB Atlas Security Capabilities

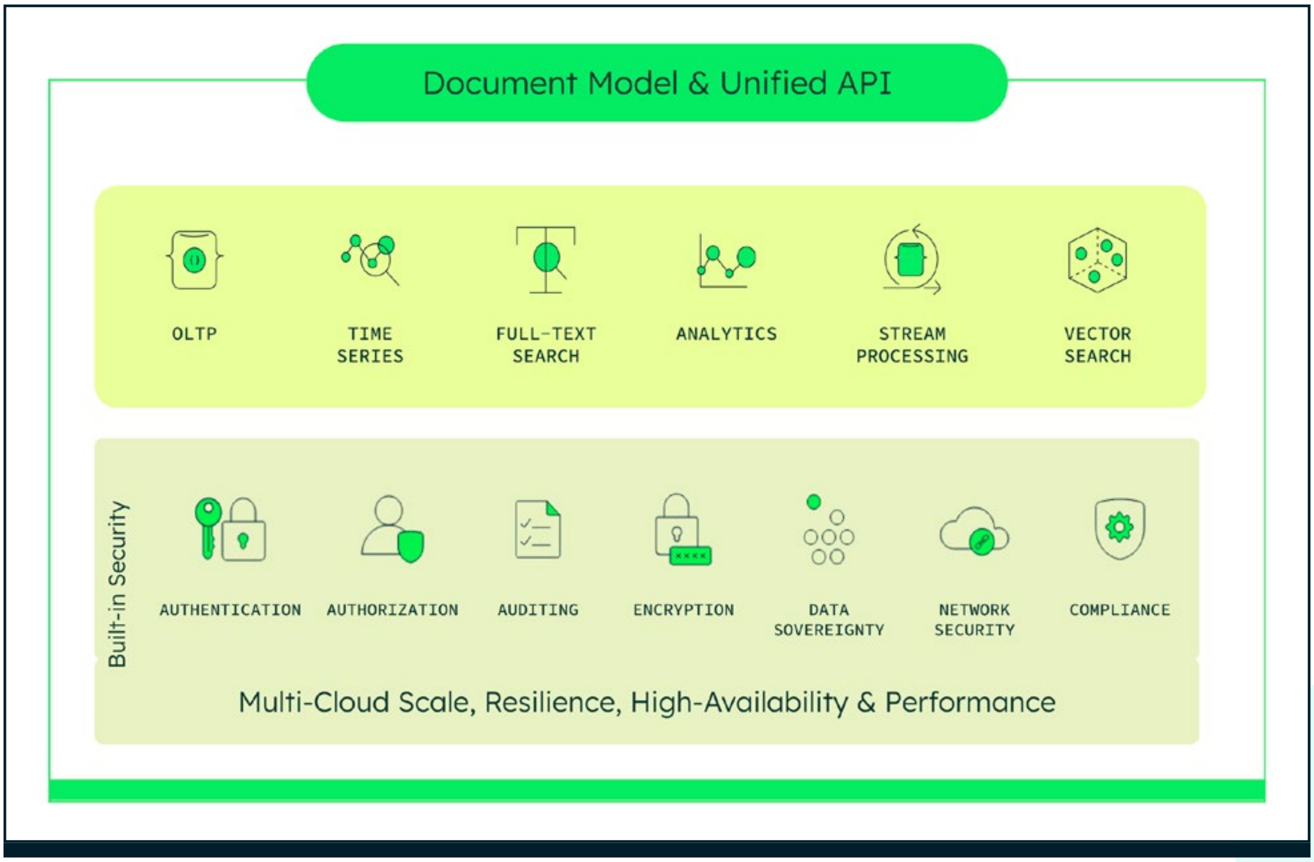


Figure 1. MongoDB Atlas Security Capabililities

MongoDB Atlas Security Capabilities at a glance	
Federated Authentication	Federated authentication using built-in integrations with Okta, Ping Identity, Azure AD, and others
Database Authentication	SCRAM, x509 certificates, AWS-IAM, LDAP
	Hashicorp Vault native integrations
Auditing	Always-on cloud user action and DB auth tracking
	Granular system activity tracking including DDL, DML, and DCL (Data Definition, Data Manipulation, and Data Control Language) commands
Encryption	MongoDB Atlas integrates with your key management services of choice – This includes integration with AWS KMS, Azure Key Vault, Google Cloud KMS for at-rest and in-use encryption (CSFLE and Queryable Encryption). Additionally, KMIP-enabled key providers can also be used with CSFLE and Queryable Encryption.
	Encryption in-use with Client-Side Field Level Encryption and Queryable Encryption
Data Sovereignty	Control data residency via cloud provider and 90+ region selection across AWS, Azure, & Google Cloud.
Network Security	IP Access lists, VPC Peering
	Private Endpoint (AWS, Google Private Service Connect, Azure Private link)
Compliance and Security Assurance	ISO 27001, 27017, 27018, CSA STAR II, SOC 2, HITRUST, PCI, VPAT, GDPR, IRAP
	FedRAMP Moderate Authorized and CJIS (MongoDB Atlas for Government)

Before we jump into details on each of the security capabilities listed above, let us quickly go through the Shared Responsibility Model.

Shared Responsibility Model

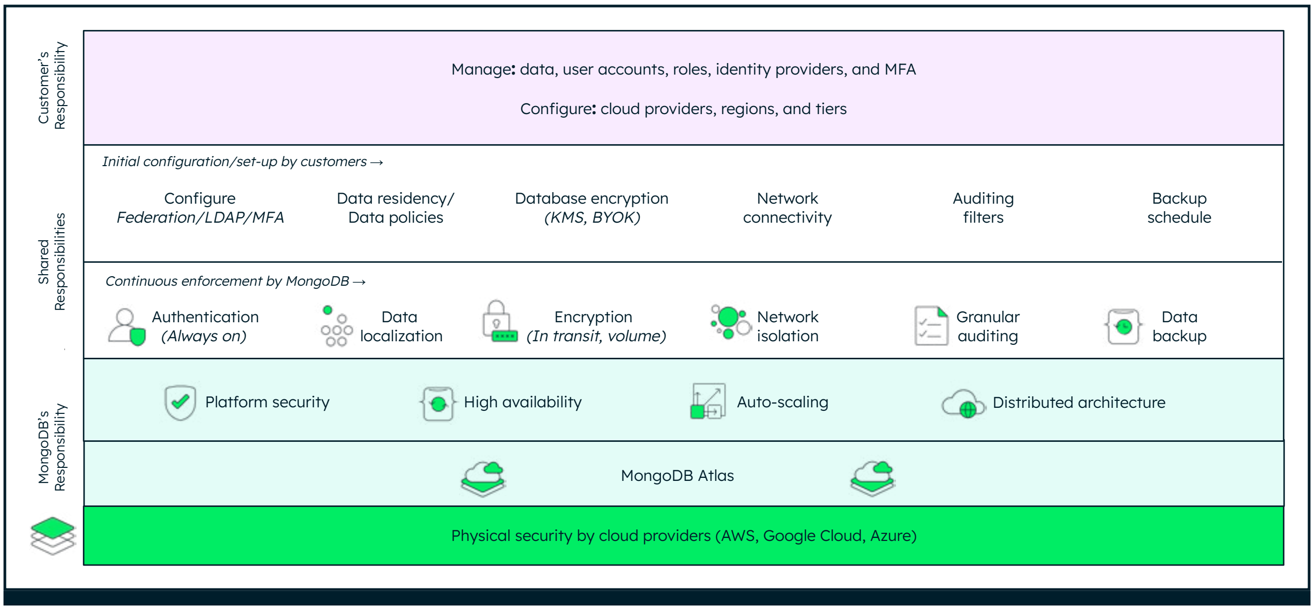

Customer's Responsibility
Manage: data, user accounts, roles, identity providers, and MFA
Configure: cloud providers, regions, and tiers
Shared Responsibilities
Initial configuration/set-up by customers →
Configure Federation/LDAP/MFA
Data residency/ Data policies
Database encryption (KMS, BYOK)
Network connectivity
Auditing filters
Backup schedule
Continuous enforcement by MongoDB →
Authentication (Always on)
Data localization
Encryption (In transit, volume)
Network isolation
Granular auditing
Data backup
MongoDB's Responsibility
Platform security
High availability
Auto-scaling
Distributed architecture
MongoDB Atlas
Physical security by cloud providers (AWS, Google Cloud, Azure)

Figure 2. The MongoDB Shared Responsibility Model

As with any cloud service, the provider and customers share responsibility for securely using the service. MongoDB Atlas has been designed with strong security defaults in mind so that the burden of securely using the service is minimized for the customer. These defaults include always-on authentication, authorization, encryption in-transit, encryption at-rest, and no database access from the internet by default. MongoDB Atlas is architected to provide automated database resilience and mitigate the downtime risks associated with hardware failures or unintended actions. For more in-depth information, read the Resilience and High Availability With MongoDB Atlas whitepaper

Customers are responsible for creating users and roles to access their MongoDB Atlas databases, selecting cloud provider(s) and region(s) in which to create their clusters and the cluster type. They can optionally enable backup, configure advanced auditing, bring their own keys for storage engine encryption, and configure in-use encryption.

For more details on the Shared Responsibility Model refer to the datasheet and the whitepaper.

Authentication and Authorization

MongoDB Atlas supports multiple authentication and authorization options and methods to give customers the flexibility to meet their individualized requirements and needs. MongoDB Atlas environment consists of a web application administrative interface (MongoDB Atlas UI) and any MongoDB Atlas Cluster you deploy. MongoDB Atlas provides you with configurable authentication and authorization options for both the MongoDB Atlas UI and your MongoDB Atlas Clusters.

The MongoDB Atlas Web UI/Control Plane is the web application where your administrators can manage Atlas clusters, including initial user and permissions setup. The MongoDB Atlas UI/Control Plane supports authentication via username/password and multi-factor authentication. Control plane user identities are managed in a MongoDB-controlled instance, encrypted and stored securely. Federated identity with SAML identity providers such as Okta or Ping Identity are supported. Users may also create and login to an Atlas control plane account using a Google Account. Authentication to the Atlas UI/Control Plane times out after 12 hours; users will need to re-authenticate after that time. For the MongoDB Atlas Cluster, authentication is automatically enabled by default to help ensure a secure system out of the box.

MongoDB Atlas allows administrators to define permissions for a user or application, and what data can be accessed when querying MongoDB. MongoDB Atlas provides the ability to provision users with roles specific to a project or database, making it possible to realize a separation of duties between different entities accessing and managing the data.

Administrators can also create temporary MongoDB users; Atlas will automatically delete the user after a specified period of time. This capability is highly complementary to granular database auditing (described in more detail below). For example, when a user needs to be granted temporary access to perform maintenance, the assigned role and all of its actions can be comprehensively audited. Once Atlas deletes the user, any client or application attempting to authenticate with the user will lose access to the database.

Multi-factor Authentication

For the MongoDB Atlas Web UI, user credentials are stored using industry-standard and audited one-way hashing mechanisms. Additionally, customers can choose to optionally utilize multi-factor authentication, or require all of the users in their Atlas Organization to use multi-factor authentication. Multi-factor authentication options include SMS, voice call, a multi-factor app, or a multi-factor device (such as a YubiKey). Customer-sensitive data provided within the GUI, such as passwords, keys, and credentials that must be used as part of the service are stored encrypted.

x.509 Authentication

Ensure tighter security controls and adhere to existing security protocols by enabling passwordless authentication to MongoDB Atlas clusters with X.509 certificates. Easily configure the X.509 option that works for your standards. X.509 is supported by two options "Easy" and "Advanced." Enable the "Easy" X.509 option in MongoDB Atlas to auto-generate certificates to authenticate your database users. If you have pre-existing certificate management infrastructure you have the ability to enable the "Advanced" X.509 option to upload your CA certificate to MongoDB Atlas and continue to use your in-house X.509 certificates for authentication. This option can be optionally combined with LDAPS for authorization. Atlas automates alerts when a certificate issued by the Atlas CA or CRL is close to expiration.

AWS IAM Authentication

Further simplifying cloud-native security, your applications, containers, and serverless functions can authenticate to MongoDB Atlas clusters reusing existing regular and temporary AWS IAM credentials. Applications provisioned on EC2 instances, Docker containers managed by ECS, or serverless functions running on AWS Lambda can automatically obtain IAM credentials from local metadata, using them to authenticate to MongoDB Atlas, just as you can for any AWS-native service. AWS IAM authentication is available only on clusters that use MongoDB version 4.4 and higher.

AWS IAM authentication is available on all Atlas clusters, including those running on other cloud providers (Google Cloud, Azure).

LDAP Integration

User authentication and authorization against MongoDB Atlas clusters can be managed via a customer's Lightweight Directory Access Protocol (LDAPS) server over TLS. A single LDAPS configuration applies to all database clusters within an Atlas project. For customers running their LDAPS server in an AWS Virtual Private Cloud (VPC), a peering connection is recommended between that environment and the VPC containing their Atlas databases.

API Access

For programmatic access to an organization or project, administrators can create organization-scoped API keys. As a prerequisite, you must turn on an organization-level setting that only allows programmatic API keys to be used if there is at least one API Access List entry. The creation and deletion of keys will be logged in the Atlas activity feed.

HashiCorp Vault Integration

You can use HashiCorp Vault to generate and manage secrets for MongoDB Atlas database users and programmatic APs, standardizing and controlling workflows with other tools and services. Two Vault secrets engines manage the life-cycle of Atlas credentials that contain a secret: the *MongoDB Atlas Secrets Engine* manages secrets for API keys, while the MongoDB Atlas Database Secrets Engine manages database users.

Auditing

Control Plane Auditing

Atlas allows administrators to audit all events triggered from the Atlas UI at the Project or Organization level. The logs are available in the Atlas UI or the API.

Always-On Database Authentication Auditing

For dedicated clusters (M10 and above), Atlas provides an easy-to-read log of database authentication events – including both successes and failures – such as database user, source IP address, and timestamp. This can be accessed either within the Atlas UI or via the API.

Granular Database Auditing

Granular database auditing in MongoDB Atlas allows administrators to answer detailed questions about systems activity by tracking all DDL, DML, and DCL commands against the database. All DML commands can be audited, including reads along with creations/updates/deletes. Admins can select the actions that they want to audit, as well as the MongoDB users, Atlas roles, and LDAPS groups whose actions they wanted to be audited, right from the Atlas UI. A single auditing configuration applies to all database clusters within an Atlas project. When needed, audit logs can be downloaded in the UI or retrieved using the MongoDB Atlas API.

Data Encryption

Data that is created, exchanged, and stored in an organization is one of its most valuable assets. Securing that data from compromise and unauthorized access, especially when it comes to personally identifiable information (PII), financial, health, or government information, should be at the very top of your priorities.

Authentication and authorization offer one level of security but your workloads which are critical to your organization have to be encrypted. MongoDB encryption offers robust features, some coming out-of-the-box on the MongoDB developer data platform, which we will cover in this article.

Encryption in Transit

All MongoDB Atlas network traffic is protected by Transport Layer Security (TLS), which is enabled by default and cannot be disabled. Customer data that is transmitted to MongoDB Atlas, as well as customer data transmitted between nodes of your MongoDB Atlas Cluster, is encrypted in transit using TLS. You can select which TLS version to use for your MongoDB Atlas Clusters, with TLS 1.2 being the recommended default and a minimum key length of 128 bits.

Key management procedures for encryption in transit

All encryption in transit is supported by the use of OpenSSL FIPS Object Module. We maintain documented cryptography and key management guidelines for the secure transmission of customer Data, and we configure our TLS encryption key protocols and parameters accordingly. MongoDB's key management procedures include:

1. Generation of keys with approved key length
2. Secure distribution, activation and storage, recovery and replacement, and update of keys
3. Recovery of keys that are lost, corrupted, or expired
4. Backup/archive of keys
5. Maintenance of key history
6. Allocation of defined key activation and deactivation dates
7. Restriction of key access to authorized individuals; and
8. Compliance with legal and regulatory requirements.

When a key is compromised, it is revoked, retired, and replaced to prevent further use (except for limited use of that compromised key to remove or verify protections). Keys are protected in storage by encryption and are stored separately from encrypted data. TLS certificates are obtained from a major, widely trusted third-party public certificate authority. In the course of standard TLS key negotiation for active sessions, ephemeral session keys are generated which are never persisted to disk, as per the design of the TLS protocol.

Encryption at Rest

Encryption at rest is a protection layer to guarantee that the written files or storage is only visible once decrypted by an authorized process/application. Upon creation of a MongoDB Atlas Cluster, by default, customer data is encrypted at rest using AES-256 to secure all volume (disk) data. That process is automated by the transparent disk encryption of your selected Cloud Provider, and the Cloud Provider fully manages the encryption keys. You may also choose to enable database-level encryption via the WiredTiger Encrypted Storage Engine (using AES-256), which allows you to bring your own encryption key with AWS Key Management Service (KMS), GCP KMS, or Azure Key Vault.

- **Amazon Web Services**
  Encryption-at-rest is automated using AWS's volume encryption, which uses industry standard AES-256 encryption to secure all volume (disk) data. All keys are fully managed by AWS.

  Customers running MongoDB Atlas may also choose to optionally enable database-level encryption for sensitive workloads via the WiredTiger Encrypted Storage Engine. This option allows customers to use their own AWS KMS, Azure Key Vault, or Google Cloud KMS keys to control the keys used for encryption at rest. This capability is described in more detail below.

- **Microsoft Azure**
  Encryption for data at rest is automated using Azure's volume encryption, which uses industry standard AES-256 encryption to secure all volume (disk) data. All keys are fully managed by Azure.

  Customers running MongoDB Atlas may also choose to optionally enable database-level encryption for sensitive workloads via the WiredTiger Encrypted Storage Engine. This option allows customers to use their own AWS KMS, Azure Key Vault or Google Cloud. KMS keys to control the keys used for encryption at rest. This capability is described in more detail below.

- **Google Cloud**
  Encryption for data at rest is automated using Google Cloud's volume encryption, which uses an Advanced Encryption Standard (AES) algorithm with a 256-bit key length, in Galois/Counter Mode (GCM). This is implemented in the BoringSSL library that Google maintains. In addition to the storage system level encryption, data is also encrypted at the storage device level with AES-256 on solid state drives (SSD), using a separate device-level key (a different key than the storage level). All keys are fully managed by Google Cloud.

Encryption at rest using Customer Key Management

Customers running MongoDB Atlas may choose to "bring their own key" and enable database-level encryption for data via the WiredTiger Encrypted Storage Engine. All Atlas databases and snapshot backups use strong volume (disk) encryption by default to protect data at rest. The use of self-managed keys with the WiredTiger Encrypted Storage Engine can help customers achieve additional levels of confidentiality and data segmentation.

Please review the Atlas documentation on Encryption Key Management for the Encrypted Storage Engine for a general overview. The following describes how customers can delegate the use of their keys.

Atlas uses a customer's unique Master Key (AWS KMS Customer Master Key, Azure Key Vault Secret Key, or Google Cloud Service Account Key) per project to generate, encrypt, and decrypt its data master keys. Master keys are then used to encrypt database keys. This process is called envelope encryption.

Encryption key concepts

MongoDB Master Key

MongoDB Master Key is an encryption key used by the MongoDB Server to encrypt the WiredTiger Storage Engine. The key isn't stored in the MongoDB database, but it's supplied externally through KMIP or a local keyfile. When the MongoDB server starts, it obtains the master key from the KMIP or local file and then stores it in memory. This key is then used to decrypt the data stored in the WiredTiger storage engine.

Atlas maintains a layer that translates requests between MongoDB Server and a CMK that you created in AWS. To translate the requests, Atlas uses the layer to request the CMK to create an encrypted data encryption key (DEK). This encrypted DEK is generated per Atlas deployment.

For example, for a three node M10+ replica set as shown in the following figure, there are three unique encrypted DEKs, one per node. Atlas stores the encrypted DEK on disk on each node in the Atlas cluster. When the cluster starts up, the Atlas layer decrypts the DEK using the customer provided encryption key and supplies this to the MongoDB Server.

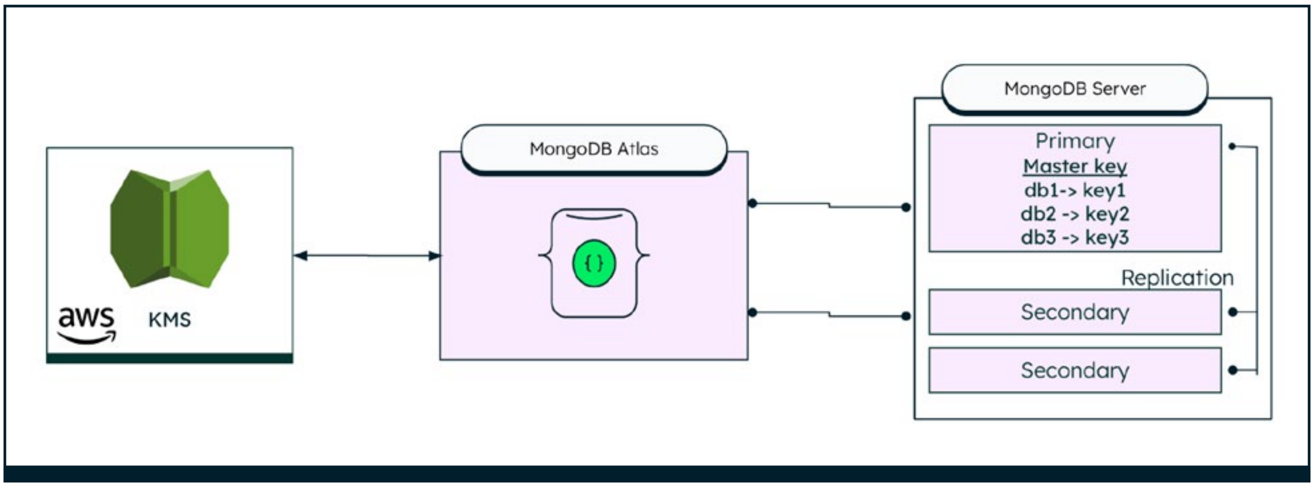


Figure 3.

Per Database Encryption Key in a MongoDB Cluster

MongoDB Server maintains a per database encryption key in the MongoDB cluster. In the preceding figure, there are three databases on the MongoDB cluster, each of which is encrypted with a unique database encryption key. Each of these keys are then encrypted with the MongoDB Master Key.

Data Encryption Key (in cloud provider terminology) or MongoDB Master Key

Atlas uses the customer provided encryption key to create an encrypted DEK. Atlas also uses a customer key management instance to decrypt this encrypted DEK and supply the resulting plaintext key to the MongoDB Server over the wire using TLS. When MongoDB Server uses this plaintext key, it refers to it as the MongoDB Master Key, whereas a cloud provider's customer key management instance might refer to it as a DEK. To learn more about DEKs, see Data Keys.

Customer Master Key (CMK)

Customer Master Key is a concept of a customer key management instance. CMKs are used to encrypt and decrypt a MongoDB Master Key (or DEK). The CMK exists only on the customer key management instance. To learn more about CMKs, see Data Keys.

The Master Key in the context of a customer's cloud service – generates and decodes data keys. When the Encrypted Storage Engine is enabled for an Atlas project, customer databases can only be started or backed up when the customer's Master Key is active and valid.

Refer to the documentation, on how to manage the master key.

Key rotation

For Encryption at-rest using customer managed keys, customers who require key rotation can use Key Management Systems (KMS) and set the master key rotation policy for automatic rotation natively inside those systems. We recommend that customers create purpose-generated IAM (Identity Access Management) profiles for access to those services, and make the scope very narrow – only encrypt/decrypt operations on a single project-specific key and isolate key administration actions (generate, delete, etc) under a management IAM profile which is inaccessible to the application.The secret keys/credentials to the encrypt/decrypt IAM service account profile itself (i.e., the authentication secrets to make an API call to the KMS) should be rotated periodically as well. Refer to the documentation for more details.

In-Use Encryption

MongoDB Atlas supports automatic encryption of individual data fields of Customer Data before they are sent to MongoDB Atlas via in-use encryption. If you enable either the Client-Side Field Level Encryption or the Queryable Encryption feature on selected data, an application-side component built into the MongoDB drivers encrypts that field before being sent to MongoDB Atlas, and only decrypts it upon return to the driver. Once encrypted, MongoDB Atlas never sees your unencrypted data. Encryption keys are managed by and only available to the application.

By securing data with in-use encryption you can move to managed services in the cloud with greater confidence. This is because the database only works with encrypted fields, and you control the encryption keys, rather than having the database provider manage the keys for you. This additional layer of security enforces an even finer-grained separation of duties between those who use the database and those who administer and manage the database.

Data encryption keys are protected by strong symmetric encryption with standard wrapping Key Encryption Keys, which can be natively integrated with external key management services backed by FIPS 140-2 validated Hardware Security Modules (HSMs). Supported key providers are Amazon KMS, Azure Key Vault, Google Cloud KMS and any KMIP-compliant key manager. As an example, customers can use remote secure web services to consume an external key or secrets manager such as Hashicorp Vault.

MongoDB's In-Use encryption features complement existing network and storage encryption to protect the most highly classified, sensitive fields of your records without

- Developers needing to write additional, highly complex encryption logic
- Significantly impacting database performance

Client-Side Field Level Encryption

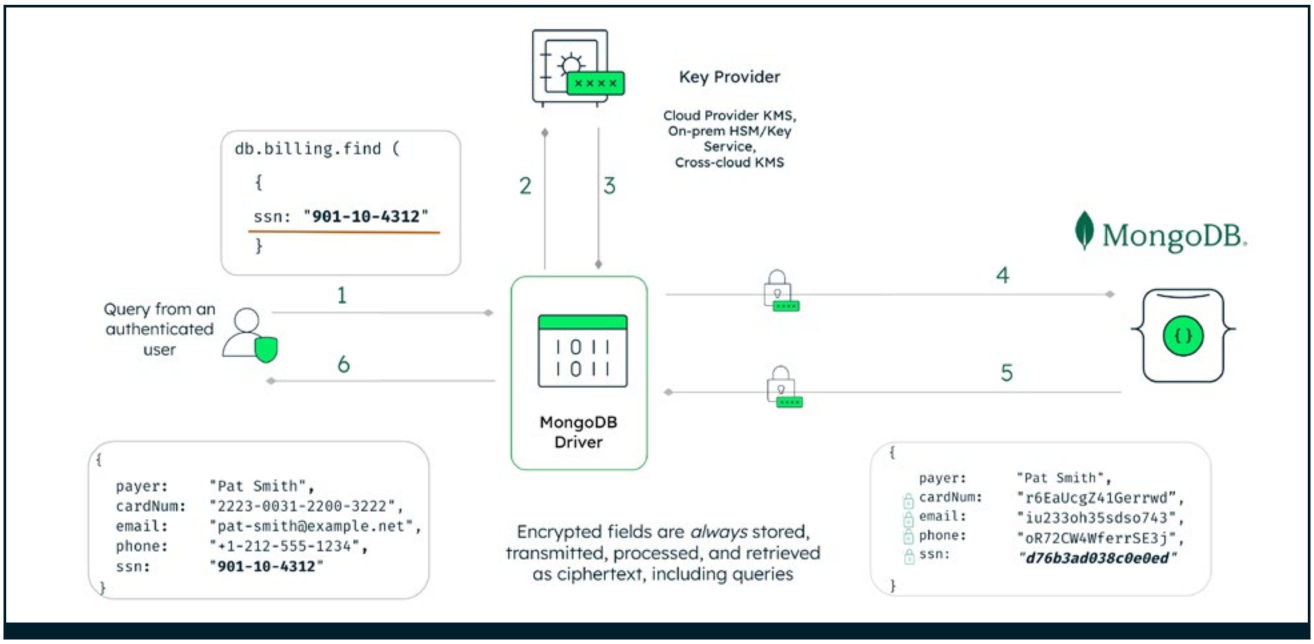
Client-Side Field Level Encryption (CSFLE) provides among the strongest levels of data privacy and security for regulated workloads. What makes Client-Side Field Level Encryption different from other database encryption approaches is that the process is totally separate from the database server. Encryption and decryption are instead handled exclusively within the MongoDB drivers in the client before sensitive data leaves the application.

CSFLE implementation

CSFLE is highly flexible. You can selectively encrypt individual fields within a document, multiple fields within the document, or the entire document. Each field can be optionally secured with its own key and decrypted seamlessly on the client.

CSFLE uses AES-256 in authenticated CBC mode: AEAD AES-256-CBC encryption algorithm with HMAC-SHA-512 MAC.

To understand how FLE works in practice, let's take a look at the flow of a query submitted by an authenticated client, as shown in Figure 4.


Key Provider
Cloud Provider KMS,
On-prem HSM/Key Service,
Cross-cloud KMS
2
3
Query from an authenticated user
1
6
MongoDB Driver
4
5
MongoDB.
Encrypted fields are *always* stored, transmitted, processed, and retrieved as ciphertext, including queries

```
db.billing.find (
  {
  ssn:  "901-10-4312"
  }
```

```
{
  payer:     "Pat Smith",
  cardNum:   "2223-0031-2200-3222",
  email:     "pat-smith@example.net",
  phone:     "+1-212-555-1234",
  ssn:       "901-10-4312"
}
```

```
{
   payer:      "Pat Smith",
   cardNum:    "r6EaUcgZ41Gerrwd",
   email:      "iu233oh35sdso743",
   phone:      "oR72CW4WferrSE3j",
   ssn:        "d76b3ad038c0e0ed"
}
```

Figure 4. MongoDB Client-Side Field Level Encryption implementation

In this example we are retrieving a user's medical record by their SSN number:

1. When the application submits the query, the MongoDB driver first analyzes it to determine if any encrypted fields are involved in the filter.
2. Recognizing that the query is against an encrypted field, the driver requests the fields' encryption key from the external key manager.
3. The key manager returns the keys to the MongoDB driver, which then encrypts the ssn field.
4. The driver submits the query to the MongoDB server with the encrypted fields rendered as ciphertext.
5. The MongoDB server returns the encrypted results of the query to the driver.
6. The query results are decrypted with the keys held by the driver, and returned to the authenticated client as readable plaintext.

Note that in the query flow, the raw key material is never persisted to disk. Rather it resides only in memory on the application server, never accessed by or transmitted to the database.

Since the database server has no access to the encryption keys, certain query operations such as sorts, regexes, and range-based queries on encrypted fields are not possible server-side. With this in mind, CSFLE is best applied to selectively protect just those fields containing highly sensitive PII such as email addresses, phone numbers, credit card information or social security numbers. Reads against fields in the document that are not encrypted client-side will evaluate as normal, as part of any query or aggregation pipeline operation.

To learn more, download our guide to CSFLE, and review these key resources:

- The Client-Side Field Level Encryption documentation provides more detail on the implementation of FLE. It covers supported encryption methods and algorithms; key management; schema enforcement, driver compatibility; and more.
- The Client-Side FLE tutorial provides worked examples in multiple languages for full stack developers using a healthcare application as an example.

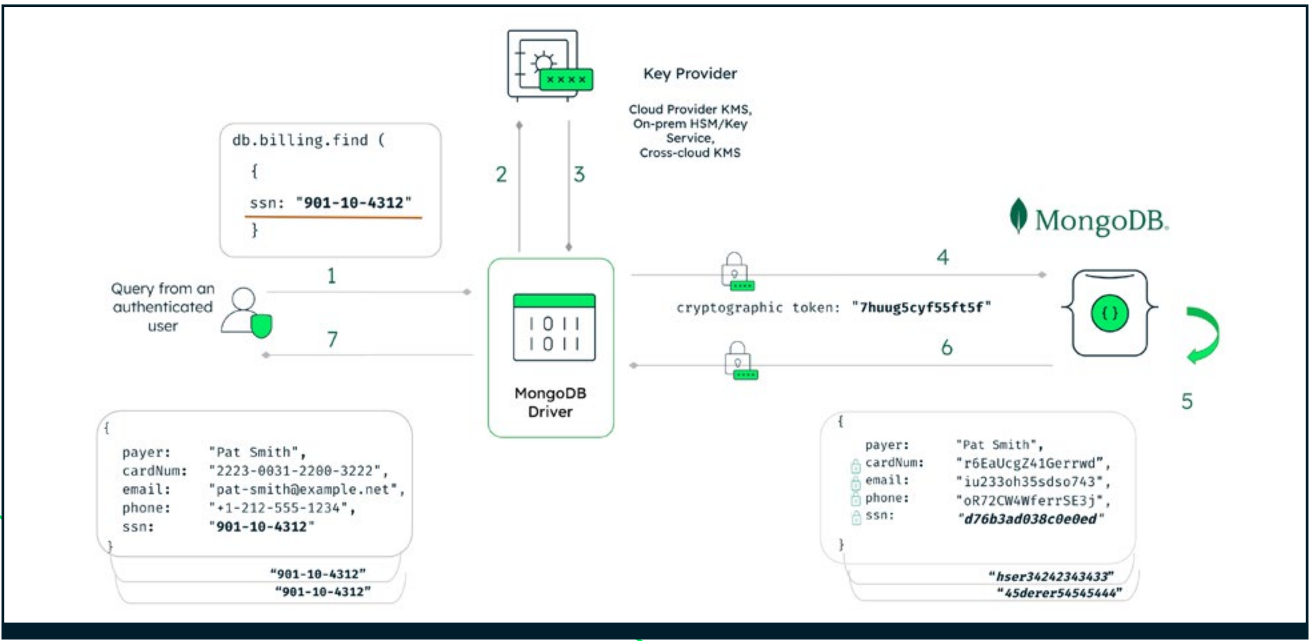
Queryable Encryption

Queryable Encryption enables an application to encrypt sensitive data from the client side using a MongoDB driver, store the encrypted data in the MongoDB database, and run expressive queries on the encrypted data using a fast state-of-the-art encrypted search algorithm.

Queryable Encryption Implementation

Queryable Encryption can be applied at the field, subdocument or document level. Data encryption is done using AES-256 in authenticated CBC mode: AEAD AES-256-CBC encryption algorithm with HMAC-SHA-256. A unique key is used for each field.

Queryable Encryption uses a deeply integrated novel in-use Structured Encryption scheme called OST and was developed by our Cryptography Research Group, backed by 20 years of academic research in the field of encrypted search.



```
db.billing.find (
  {
  ssn: "901-10-4312"
  }
```

```
cryptographic token: "7huug5cyf55ft5f"
```

```
{
  payer:     "Pat Smith",
  cardNum:   "2223-0031-2200-3222",
  email:     "pat-smith@example.net",
  phone:     "+1-212-555-1234",
  ssn:       "901-10-4312"
}
           "901-10-4312"
           "901-10-4312"
```

```
{
  payer:      "Pat Smith",
  cardNum:    "r6EaUcgZ41Gerrwd",
  email:      "iu233oh35sdso743",
  phone:      "oR72CW4WferrSE3j",
  ssn:        "d76b3ad038c0e0ed"
}
            "hser34242343433"
            "45derer54545444"
```

Figure 5. Queryable Encryption flow of operations

1. Authorized users run an equality query to get specific SSN number records
2. Recognizing the query is against an encrypted field, the driver requests the encryption keys from the customer-provisioned key provider, such as AWS Key Management Service (AWS KMS), Google Cloud KMS, Azure Key Vault, or any KMIP-enabled provider, such as HashiCorp Vault.
3. The MongoDB driver gets the encryption keys from the key provider
4. The driver submits the encrypted query along with a cryptographic token to the MongoDB server with the encrypted fields rendered as ciphertext.
5. Queryable Encryption implements a fast encrypted search algorithm that allows the server to process queries on the encrypted data, without knowing the data. The data and the query itself remain encrypted at all times on the server.
6. The MongoDB server returns the encrypted results of the query to the driver.
7. The query results are decrypted with the keys held by the driver and returned to the client and shown as plaintext.

Note that in the query flow, the raw key material is never persisted to disk. Rather it resides only in memory on the application server, never accessed by or transmitted to the database.

Currently equality query is supported (as of 7.0 release) and additional query types such as range, prefix, suffix, and substring will be added in future releases.

To learn more, review these key resources:

- The Queryable Encryption documentation provides more detail on the implementation of Queryable Encryption. It covers supported encryption methods and algorithms; key management; schema enforcement, driver compatibility; and more.
- The Queryable Encryption tutorial provides worked examples in multiple languages for full stack developers using a healthcare application as an example.

Both CSFLE and Queryable Encryption are supported and will continue to be for the foreseeable future. Client-Side Field Level Encryption provides high levels of confidentiality for sensitive data and is a strong solution for most workloads, including the ability to run equality queries on deterministically encrypted data. Customer-controlled keys scoped at the application ensure that only the application - not a DBA, not a cloud provider, and not a 3rd party service provider - has access to the data in its unencrypted form. Queryable Encryption has all of the security guarantees of CSFLE and adds the ability to run equality queries on randomized encrypted data with more query types to come.

This table compares the key benefits of CSFLE and Queryable Encryption.

Key Benefits	Queryable Encryption	CSFLE
Faster application development cycle Developers don’t have to figure out how to use the right algorithms, encryption options, etc to implement their right encryption solution. MongoDB has done all that complex work for them.	✓	✓
Strong technical controls for critical data privacy use cases Help customers meet strict data privacy requirements such as HIPAA, GDPR, CCPA, PCI, and more.	✓	✓
End–to–end encryption Data is encrypted at the client-side, remains encrypted in-transit, at rest, and in-use, and is only decrypted back at the client. Fully randomized encryption means that a given value encrypts to a different ciphertext every time.	✓	✓ *
Reduce operational risk Eliminate common security concerns when moving database workloads to the cloud. Customers can keep their data on any of the cloud providers and be assured that their data is protected.	✓	✓
Robust key management Rotate keys and migrate from one key provider to another seamlessly, without impact to your application.	✓	✓
Groundbreaking query technology Queryable Encryption introduces a fast state-of-the-art encrypted search algorithm using NIST standards-based cryptographic primitives such as AES-256, SHA2, and HMACs.	✓	
Rich querying capabilities on encrypted data Data can be queried using equality matches (generally available) with range, prefix, suffix, and substring query capabilities planned.	✓ **	

* Randomized encryption is only available with a non-searchable option.

** Currently equality query type is supported but in the future other query types like range, prefix, suffix, and substring will be added

Data Sovereignty

Data Sovereignty is the idea that data are subject to the laws and governance structures of the region where they are collected. The concept of data sovereignty is closely linked with data residency, data security, cloud computing, network security, and technical controls. For example, what may be deemed as the acceptable use of personal information in one geography would not be so in some other region. To avoid data residency compliance issues, users need to conduct data mapping – that is, understanding what data you have, where it's located, and the data residency policies for each respective location.

While data protection regulations such as GDPR, CCPA, HIPAA, PCI-DSS, and others stipulate requirements that are unique to specific regions, industries or applications, there are foundational requirements common across all of the directives, including:

- Physical storage of data in a particular geography or lack of explicit guidance
- Restricting processing of data stored in a particular geography outside that geo
- Restricting data access, enforced via user privileges and roles
- The separation of duties when accessing and processing data.
- Recording user, administrative staff, and application activities with a database.
- Ability to remove personal data when requested.
- Measures to protect against accidental or malicious disclosure, loss, destruction, or damage of personal data

MongoDB provides two mechanisms to meet with data sovereignty requirements:

1. **Dedicated MongoDB clusters in a cloud provider and region of customer's choice**

   Ability to store data in any of the regions across AWS, Google Cloud and Azure as per your data locality requirements

   Atlas databases are available in 90+ regions across AWS, Google Cloud, and Azure. You can even take advantage of multi-cloud and multi-region deployments, allowing you to target the providers and regions that best serve your users. Best-in-class automation and proven practices guarantee availability, scalability, and compliance with the most demanding data security and privacy standards. To support the data sovereignty requirements of modern privacy regulations, MongoDB zones allow precise control over where personal data is physically stored in a cluster. Zones can be configured to automatically "shard" (partition) the data based on the user's location – for example enabling administrators to isolate personal data to just countries in the EU. If a company or regulatory policies towards storing data in specific regions change, updating the shard key range enables the database to automatically migrate personal data to alternative regions. With MongoDB Atlas, you can control your data residency the way you desire – Single region or multi-regions and Single cloud or multi-cloud deployments.

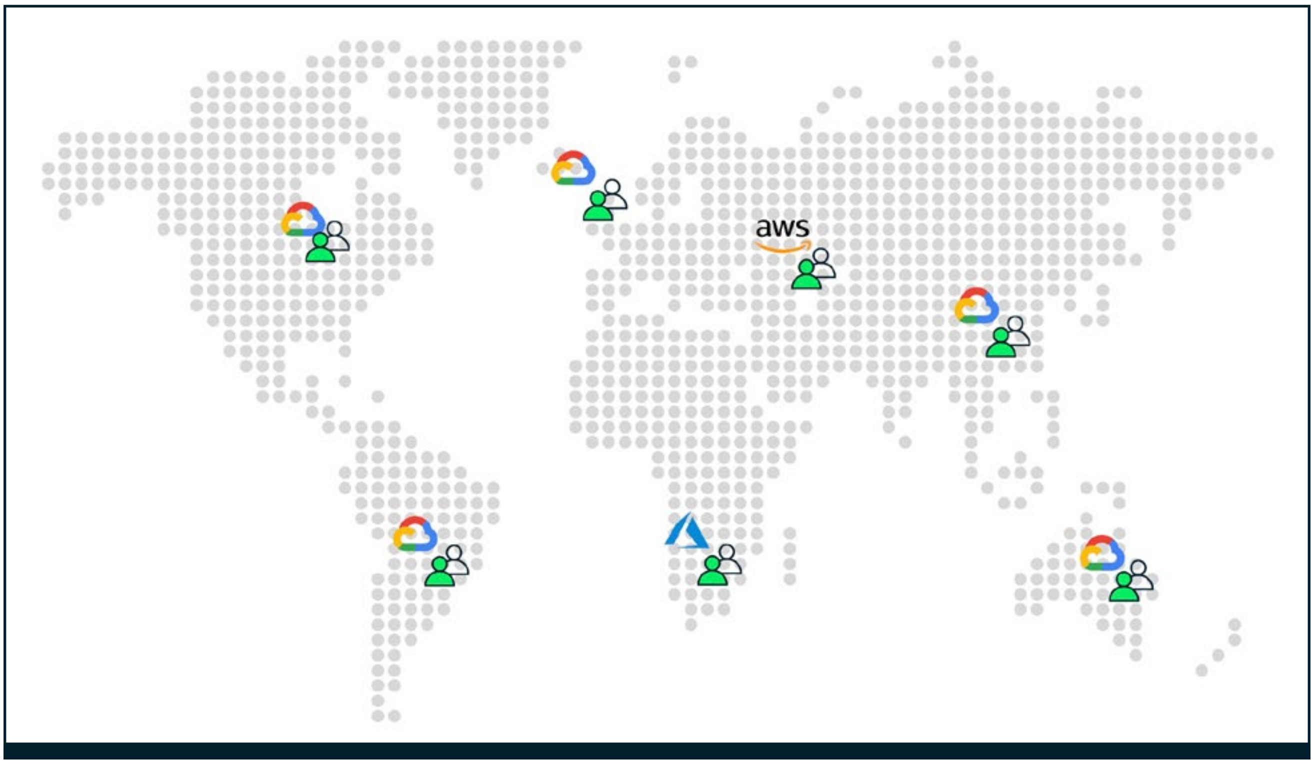


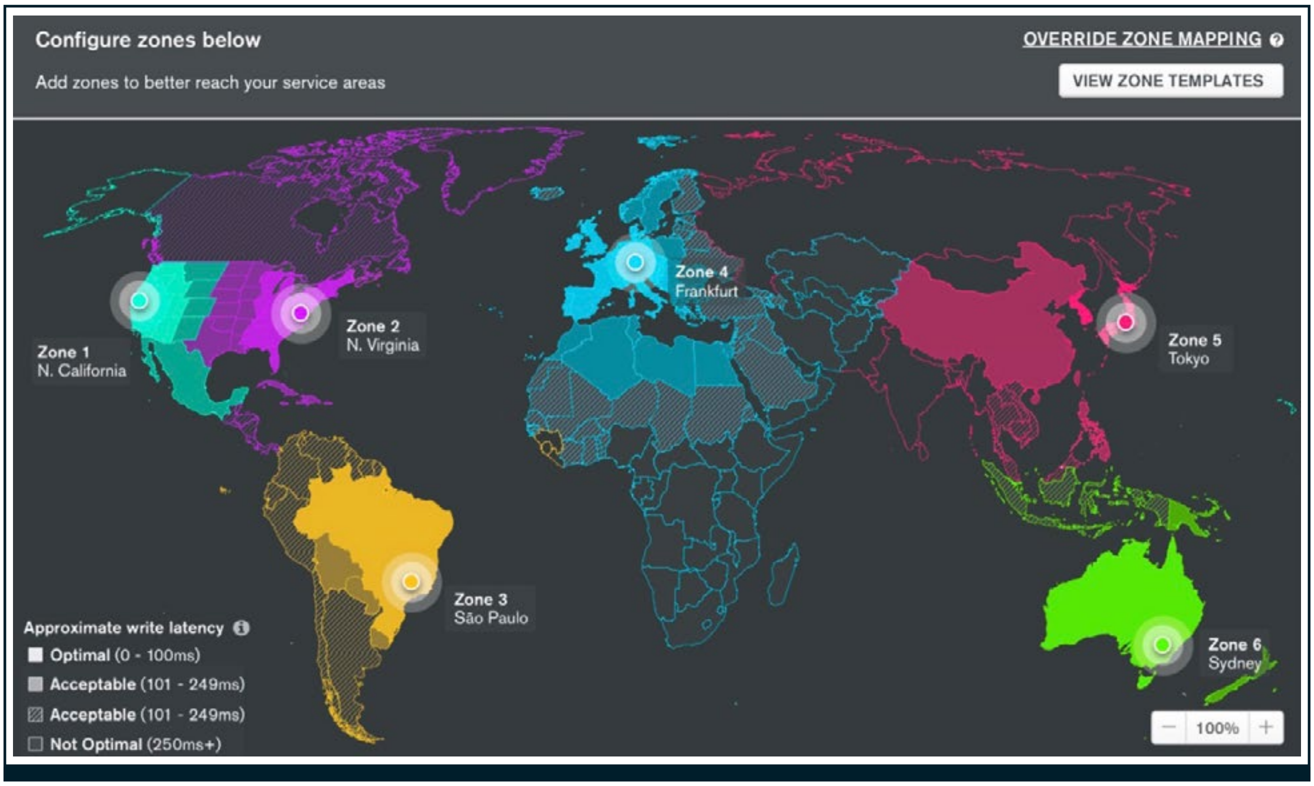
Figure 6. MongoDB Atlas Clusters can be spread across many regions

2. **Global clusters with zoned sharding**

Zoned sharding is available to MongoDB Atlas customers as part of the Global Clusters that are fully-managed cloud service providing a highly curated implementation of zoned sharding to support location-aware storage and database operations for globally distributed application instances and clients. For more information on global clusters refer to the documentation.

In addition, MongoDB offers strict security controls with features discussed above like authentication, authorization, auditing, encryption, and network security along with proactive monitoring of our platform. All of this help to support customers' data residency requirements.

MongoDB Atlas undergoes independent verification of platform security, privacy, and compliance controls. From the perspective of the GDPR, MongoDB Atlas is GDPR compliant. MongoDB and the Atlas service is classified as data processor. MongoDB's terms of service reflect the GDPR's requirements, whereby we implement the appropriate technical and organizational measures in such a manner that processing will meet Regulation requirements and protect against destruction, loss, alteration, and unauthorized disclosure or access to personal data. You can learn more about MongoDB Atlas GDPR compliance from the MongoDB trust center.



Configure zones below

Add zones to better reach your service areas

Figure 7. MongoDB Atlas Global Clusters with Zoned Sharding

Network Security

Connectivity

MongoDB requires the following network ports for Atlas. Network ports cannot be changed.

- 27017 for mongod (database server)
- 27016 for mongos (query router for sharded clusters)
- 27015 for the BI connector
- If LDAPS is enabled, MongoDB requires LDAPS network port 636 on the customer side open to inbound traffic by Atlas

For detail configuration settings, please refer to the network and firewall settings.

You can connect to Atlas via either public IPs (which are protected with IP Access Lists, discussed below) or private IPs (via network peering or private endpoints, discussed below). Connection method for public vs. private IPs varies between cloud providers, as discussed in the following sections.

Atlas cluster public IPs remain the same in the majority of cases of cluster changes: vertical scaling, topology changes, maintenance events, healing events, etc. However, certain topology changes – such as a conversion from a replica set to a sharded cluster, an addition of shards, or a region change – will require new IP addresses to be used.

IP Access Lists

By default, your MongoDB Atlas cluster will have no database access from the internet. Each Atlas cluster is deployed within a VPC configured to allow no inbound access by default.

Customers can configure IP Access Lists to limit which IP addresses can attempt to authenticate to their database. Application servers are prevented from accessing the database unless their IP addresses (or a CIDR covering their IP addresses) have been added to the IP Access List for the appropriate MongoDB Atlas project.

Atlas also supports creating temporary access list entries that automatically expire within a user-configurable period. This can be useful in situations when a member of the team needs access to an environment from a temporary work location.

As a general best practice to reduce the attack surface, MongoDB recommends customers only permit IP access to the smallest network segments possible (e.g., individual /32 address), and to avoid overly large CIDR blocks.

Network Peering

Network peering allows you to connect your own VPCs with an Atlas VPC, routing traffic privately and remaining isolated from the public internet. When you set up network peering, you can choose to only enable access via private IP from the peered network(s), or also allow access via public IP (controlled by the IP Access List).

Atlas does not need access into peered VPCs except when LDAPS is enabled. In that scenario, Atlas clusters need to reach the customer's LDAPS directory inside their VPC using the LDAPS protocol.

Customers worried about peering extending the network trust boundary to their dedicated Atlas-side VPCs can set up mitigating controls, including security groups and network ACLs, to not allow any inbound access to instances in their VPC from the Atlas-side VPC.

Customers with legacy VPCs internally that contain a large amount of infrastructure without isolation may be particularly uncomfortable introducing VPC peering and associated access governance. These customers should deploy net new VPCs for the applications requiring access to Atlas, isolating resources from each other within their own organizational network. These new VPCs can in turn be peered with the legacy/central VPCs.

Applications inside of such a VPC can reach both Atlas and other internal services but since VPC peering is non-transitive, Atlas cannot reach beyond the directly peered VPC – i.e., Atlas cannot reach your central VPCs. AWS Transit Gateway and AWS Direct Connect do provide transitive connectivity, so customers using AWS PrivateLink can use Transit Gateway or Direct Connect with your VPC to connect to Atlas via AWS PrivateLink (FAQ).

Private Endpoints

This connection method uses a one-way connection from your own VPC to the Atlas VPC. Atlas VPCs can't initiate connections back to your VPCs, ensuring that your network trust boundary is not extended.

Connections to private endpoints within your VPC can be made transitively from

- Another VPC peered to the private endpoint-connected VPC.
- An on-premises data center connected with DirectConnect to the private endpoint-connected VPC.

Private endpoints are available on:

- AWS via AWS PrivateLink
- Azure via Azure Private Link,
- Google Cloud via Private Service Connect

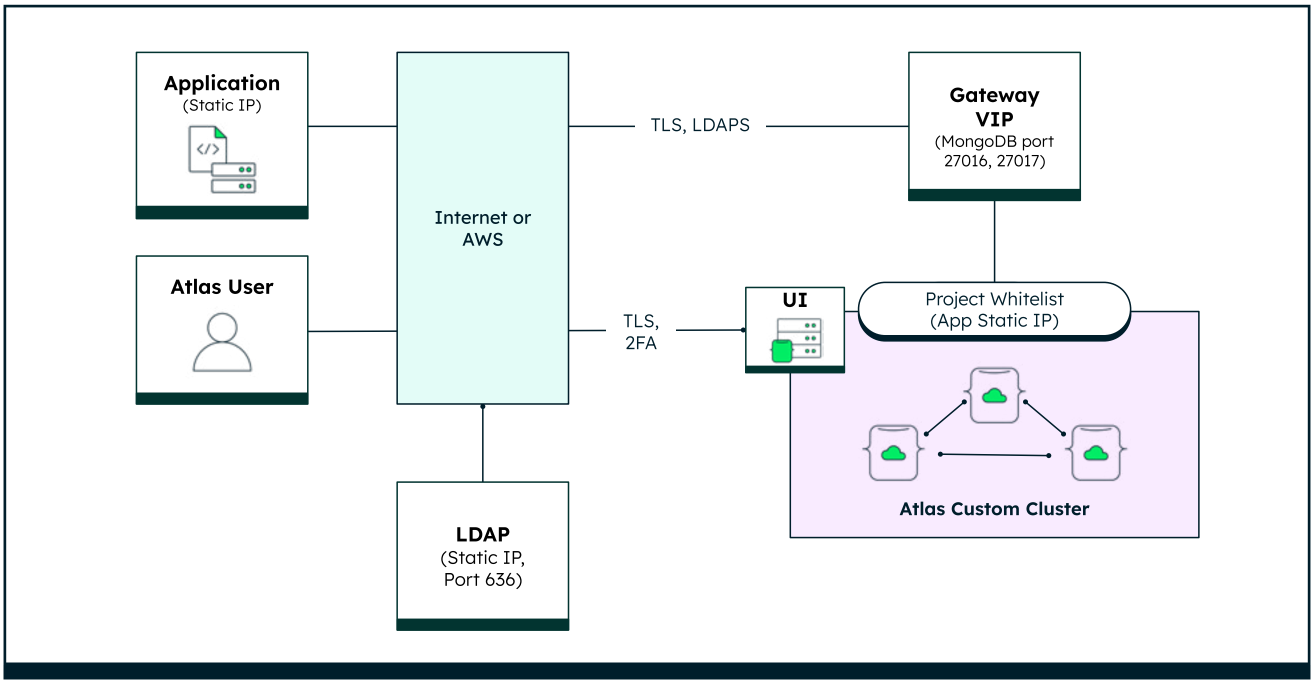
AWS VPC Topology

This section helps you review common practices to securely connect your individual clients to a MongoDB Atlas service running in an Amazon Web Services Virtual Private Cloud (VPC).

Atlas deploys a cluster in a dedicated AWS VPC and then uses authentication and the IP Access List to isolate the service. On AWS, a cross-region cluster will span multiple VPCs and an Atlas project with clusters in different regions will be using a VPC per region.

If leveraging VPC peering, the AWS VPC resolves hostnames in an Atlas cluster to their private IP addresses when you enable DNS resolution. You can use these DNS entries to connect to hosts in your Atlas cluster from the peered VPC since AWS handles resolving the peered hostnames automatically.

Single-region VPC peering connections enable Atlas to reference security groups in the peered VPC by security group ID. Atlas also supports leveraging cross-region VPC peering connections. When doing so, it is not possible to reference security groups in a peered VPC on the Atlas Access List.



Customers leveraging custom DNS solutions that cannot take advantage of built-in split-horizon DNS may enable a project setting that provides a connection string that will resolve only to private IPs.

An additional networking option for AWS is AWS PrivateLink. With PrivateLink, Atlas clusters cannot initiate connections back to your application VPC, preserving your network trust boundary and reducing your security risk. AWS PrivateLink simplifies your network architecture by allowing you to use the same set of security controls across your organization. It also provides transitive connectivity from other peered and Direct Connect contexts, allowing you to connect to Atlas locally and from on-prem data centers without using public IPs via the IP Access List.

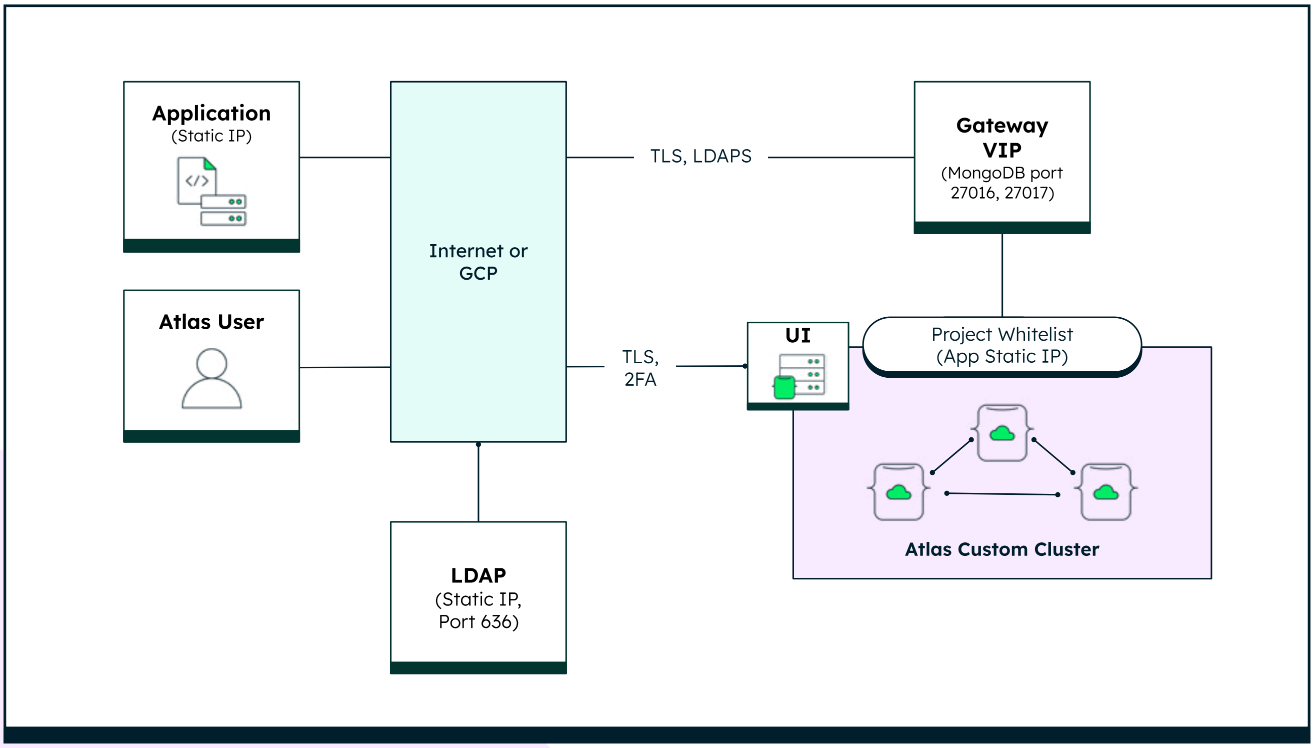
Google Cloud VPC Topology

This section helps you review common practices to securely connect your individual clients to a MongoDB Atlas service running in a Google Cloud VPC.

Atlas deploys a cluster in a dedicated global Google Cloud VPC and then uses authentication and the IP Access List to isolate the service. A logical service in Google Cloud has its DNS name registered upon creation. The DNS name points to a gateway virtual IP (VIP) address in the datacenter where the service was created. Your individual application client needs a static IP assigned, which gets added to the project access list in Atlas.

VPC peering is available for MongoDB Atlas deployments on Google Cloud. Once enabled, users can choose to connect to their MongoDB Atlas cluster either with public IPs added to the Access List or VPC peering connections.

On Google Cloud, a cross-region cluster will use a single VPC, and an Atlas project with clusters in different regions will also use a single VPC.

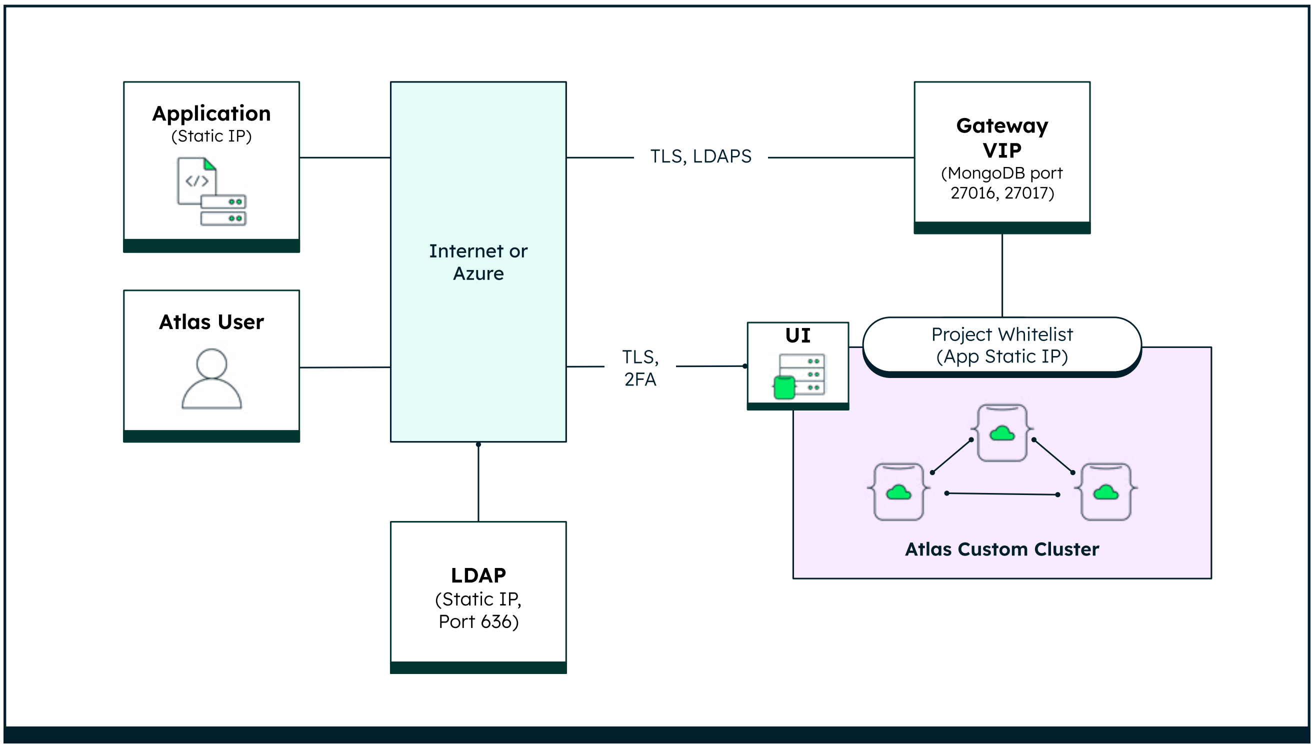


Microsoft Azure VNET Topology

This section helps you review common practices to securely connect your individual clients to a MongoDB Atlas service running in an Azure Virtual Network (VNet).

Atlas deploys a cluster in a dedicated Azure VNet and then uses authentication and the IP Access List to isolate the service. A logical service in Microsoft Azure has its DNS name registered upon creation. The DNS name points to a gateway virtual IP (VIP) address in the datacenter where the service was created. Your individual application client needs a static IP assigned, which gets added to the project access list in Atlas.

On Azure, a cross-region cluster will span multiple VNets and an Atlas project with clusters in different regions will be using a VNet per region.



VNet peering is available for MongoDB Atlas deployments on Azure, for both the single region and multi-region clusters. Once enabled, users can choose to connect to their cluster either with public IPs via the Access List or VNet peering connections.

An additional networking option for Azure is Azure Private Link. With Private Link, Atlas clusters cannot initiate connections back to your application VNet, preserving your network trust boundary and reducing your security risk. Azure Private Link simplifies your network architecture by allowing you to use the same set of security controls across your organization. It also provides transitive connectivity from other peered and ExpressRoute contexts, allowing you to connect to Atlas locally and from on-prem data centers without using public IPs via the IP Access List.

Compliance & Trust

MongoDB has a comprehensive compliance & trust program for its cloud offerings. MongoDB is committed to delivering the highest levels of standards conformance and regulatory compliance as part of our ongoing mission to address the most demanding security and privacy requirements of our customers.

MongoDB Atlas Compliance and Attestations

The scope of services under compliance and trust includes Atlas Database, Atlas Search, Atlas Data Lake, Charts, MongoDB Realm, Cloud Manager, and MongoDB Serverless.

ISO 27001

The ISO/IEC 27001 family of standards is designed to help manage the global security of assets such as financial information, intellectual property, employee details or information entrusted to a service provider. Today there are more than a dozen 27000 family standards. 27001 sets requirements for an information security management system (ISMS). MongoDB cloud services has achieved ISO/IEC 27001:2013 certification. Learn More.

ISO 27017

ISO/IEC 27017:2015 provides guidance and recommendations of implementing cloud-specific information security controls that supplement the ISO/IEC 27001 standards, to ensure continuous management of security in a comprehensive manner. Learn More.

ISO 27018

ISO/IEC 27018:2019 is one of the critical components of cloud security – protecting data. There is sensitive data on the cloud, especially personally identifiable information (PII), company proprietary, and other sensitive data which is important to protect for organizations. ISO 27018 standard focuses on security controls that are built upon existing ISO/IEC 27002 security controls and provides new controls for personal data protection. Learn More.

SOC 2

Service Organization Controls (SOC) framework establishes a standard for controls that safeguard the confidentiality and privacy of information stored and processed in the cloud. MongoDB Atlas is audited at least annually against the SOC reporting framework by independent third-party auditors. The audit covers controls for data security; the report is available to customers who've signed an NDA with MongoDB, Inc. Learn More.

PCI DSS

The Payment Card Industry Data Security Standard (PCI DSS) applies to all entities that store, process, and/or transmit cardholder data. MongoDB Atlas has been validated as a PCI DSS v4 compliant service provider by an independent Qualified Security Assessor (QSA). Customers are still responsible for managing the security of their own PCI DSS certification as well as configuring their MongoDB Atlas deployments to comply with their PCI DSS requirements. Learn More.

HIPAA

For customers who are subject to the requirements of the Health Insurance Portability and Accountability Act of 1996, MongoDB Atlas supports HIPAA compliance and enables covered entities and their business associates to use a secure MongoDB Atlas environment to process, maintain, and store protected health information. MongoDB, Inc. will enter into Business Associate Agreements covering MongoDB Atlas with customers as necessary under HIPAA. Learn More.

HITRUST

MongoDB maintains a SOC 2 + HITRUST certification report, mapping MongoDB's SOC 2 Type II controls to the 75 required HITRUST controls for certification. Mapping requirements between SOC 2 and HITRUST is an approach recommended by both AICPA (SOC) and HITRUST. Learn More.

GDPR

The General Data Protection Regulation (GDPR) standardizes data protection law across all 28 EU countries and imposes strict new rules on controlling and processing personally identifiable information. The terms of service applicable to MongoDB Atlas automatically include data processing protections that satisfy the requirements that the GDPR imposes on data controllers' relationships to data processors. Learn More.

CSA STAR

MongoDB has achieved CSA STAR Level 2, via a third-party audit of Atlas's security. The CSA Security, Trust, Assurance, and Risk (STAR) Registry is a publicly accessible registry that documents the security and privacy controls provided by popular cloud computing offerings. STAR encompasses the key principles of transparency, rigorous auditing, and harmonization of standards outlined in the CSA's Cloud Controls Matrix (CCM). Learn More.

VPAT

MongoDB has issued Accessibility Conformance Reports based on VPAT for MongoDB Atlas, MongoDB Atlas for Government, and the MongoDB database software. Download MongoDB's VPAT reports. Learn More.

IRAP

IRAP assessment is a comprehensive cybersecurity assessment framework of the security capabilities of cloud service providers, ensuring adherence to the highest security standards in order to assist Australian government agencies and departments in protecting their information and communications technology (ICT) systems from potential cyber threats. Learn More.

MongoDB Atlas for Government Compliance

MongoDB Atlas for Government (US) is a FedRAMP® authorized and dedicated environment of MongoDB Atlas for the US public sector as well as ISVs looking to build offerings for the US public sector.

FedRAMP® Moderate Authorized

FedRAMP® is a government-wide program that provides a standardized approach to security assessment, authorization, and continuous monitoring for cloud products and services. MongoDB Atlas for Government is FedRAMP Moderate Authorized. Atlas for Government is an independent, dedicated environment for the US public sector, as well as ISVs looking to build US public sector offerings. This platform is operated by MongoDB employees who are U.S persons on U.S soil – is an integrated set of data and application services that share a unified developer experience – supports a wide range of use cases including transactional workloads, time series data, search, and petabyte data storage. Learn More. Documentation.

Criminal Justice Information Solutions (CJIS)

There is no standardized accreditation or assessment for CJIS compliance. There are set security standards and controls laid out in the CJIS Security Policy and MongoDB is committed to helping customers meet those requirements. Additionally, MongoDB engaged an independent auditor to evaluate how MongoDB Atlas for Government (US) aligns with CJIS requirements. This attestation letter is available to customers subject to CJIS requirements by request. Learn More.

Business Continuity and Disaster Recovery

MongoDB maintains a formal business continuity and disaster recovery process which covers its RTO[1] and RPO[2] with regard to its Atlas control plane, and the supporting infrastructure of customer clusters in Atlas including VMs, DNS, and logs.

The Atlas control plane is the controller that provisions MongoDB clusters in one or more regions as requested by the customers. Once the Atlas control plane creates MongoDB database clusters, these clusters can continue to operate even if the control plane infrastructure is offline. If the supporting infrastructure of customer clusters is unavailable, customer connectivity to Atlas clusters may be impaired (e.g., due to DNS name resolution failure), or certain functionality may be impaired (e.g., storing and downloading logs).

Atlas Control Plane

The Atlas Control Plane is the controller accessible via MongoDB Cloud Services that provisions MongoDB clusters in one or more cloud provider regions as requested by the customer. The Atlas Control Plane is deployed in AWS in North America. Once the Atlas Control Plane creates MongoDB database clusters, these clusters can continue to operate and are accessible even if the control plane is not accessible.

Consequences of downtime for the Atlas Control Plane

If the Atlas Control Plane becomes completely inaccessible, i.e., a customer cannot login into the Atlas Control Plane UI and API, the following functionalities are unavailable.

1. Configuration
   - Provisioning new deployments or modifying existing deployments
   - Changing Project configuration, i.e., BYOK, auditing, creating dataplane users
   - Changing backup configuration
2. Non-deployment Access
   - Scheduling of automated backups
   - Data Explorer, Performance Advisor, Schema Advisor
   - Download of deployment logs and audit logs
   - Access to Activity Feed
3. Alerting
   - Alerts would not fire if the Atlas Control Plane was unavailable.

Even if the Atlas control plane is fully available, the provisioning of new deployments or modification of existing deployments depends on the underlying capacity in the Atlas Data Plane regions.

Atlas Data Plane

The Atlas data plane refers to the MongoDB clusters provisioned by Atlas on behalf of customers. The clusters running on tier M10 or above are deployed in a three node configuration in three availability zones (if available) within a cloud provider region. Availability zones are geographically distributed datacenters located far enough apart to protect each datacenter from regional disturbances.The SLA of the deployed clusters is defined here.

Each node in the deployment has a hostname. Atlas uses Route53 for the deployment hostnames. Route53 is a global AWS Service that has both control and data plane. The data plane is available in over 200 locations. Please refer to this AWS link for more information.

How can you achieve your own business continuity objectives with MongoDB Atlas?

Customers have a responsibility to maintain their own business continuity/disaster recovery and define their own RTO[1]/RPO[2] values according to their acceptable criteria (e.g., RTO/RPO of 0-4 hours), which can be achieved independently of the MongoDB Atlas control plane RTO/RPO, via the use of specific product features available to customers. These features include:

- Selection of the underlying cloud provider(s) – AWS, Google Cloud, Azure – for deploying MongoDB clusters, in order to mitigate the risk of a cloud provider failure.
- Selection of one or more cloud provider(s) regions, in order to mitigate the risk of a region failure.
- Selection of a clustered tier – shared or dedicated, sharded or unsharded – to mitigate the impact of workload spikes.
- Selection of network connectivity options to Atlas for high availability (learn more →)
- Selection of backup & restore options and the backup schedule.

Infrastructure Service Recovery

MongoDB Atlas creates and configures dedicated clusters on infrastructure provided by AWS, Azure and/or Google Cloud. Data availability also is subject to the infrastructure provider service Business Continuity Plans (BCP) and Disaster Recovery (DR) processes. Our infrastructure service providers hold a number of certifications and audit reports for these controls. For more information, please see below:

- Amazon Web Services Compliance
- Microsoft Azure Compliance
- Google Cloud Compliance

Cloud Backup

Available for Atlas clusters deployed in Amazon Web Services, Microsoft Azure, and Google Cloud, cloud provider snapshots use the native snapshot capabilities of the underlying cloud provider.

Backups are stored in the same cloud region as the corresponding cluster. For multi-region clusters, snapshots are stored in the cluster's preferred region. All managed snapshots and images are automatically encrypted. If the encryption key management integration with AWS KMS, Azure Key Vault, or Google Cloud KMS is enabled, your AWS Customer Master Key (CMK) /Azure Key Vault Secret Key / Google Cloud Service Account Key and IAM credentials are required to perform restores of backup snapshots. Cloud Backup enables you to customize the snapshot schedule and retention policies, with support for multi-year retention, making it easier for you to adhere to compliance obligations. An optional add-on, Continuous Cloud Backup, records the oplog for a configured window, permitting a restore to any point in time within that window and satisfying Recovery Point Objectives (RPOs) as low as 1 minute.

[1] RTO: Recovery Time Objective–describes how long it will take to get an application back online

[2] RPO: A Recovery Point Objective–is the maximum amount of data that can be lost before causing detrimental harm to the organization

Incident Response

The Corporate Security team employs industry-standard diagnostic procedures to drive resolution during business-impacting events. Staff operators provide 24x7x365 coverage to detect incidents and manage the impact and resolution.

Resiliency Plans

MongoDB's Corporate Security group has reviewed the MongoDB resiliency plans, which are also periodically reviewed by members of the Senior Executive management team.

Support Coverage

For customers who have purchased an Atlas support plan, the MongoDB Technical Services Engineering team provides support for the GA releases of the following software:

- MongoDB Server
- MongoDB Cloud Manager
- MongoDB Atlas
- MongoDB Atlas Search
- MongoDB Atlas Data Lake
- MongoDB Compass
- MongoDB Charts
- MongoDB Realm

Support is also provided to the tools and integrations pertaining to usage of the Atlas products including:

- MongoDB Drivers
- MongoDB Connectors, including BI and Spark
- Authentication/access controls to the Atlas clusters
- AWS, Azure, and Google Cloud integration related questions
- Performance
- Data Migrations

Refer to these support policies for more information:

- MongoDB Atlas for Government Support Policy
- MongoDB Cloud Services Support Policy

Platform – Infrastructure and Data Security

MongoDB Atlas's infrastructure is designed to be fully automated via modern configuration management systems. Reducing human elements increases a security posture by reducing the chance for human error and making audit and alerting standardized. MongoDB Atlas provisions Virtual Machines with hardened machine images built in-house, and all of our virtual servers are configuration-managed using Chef, which includes hardening steps. All systems run with a known set of running processes/components, which in turn is utilized for update/patching.

Separation of Production and Non-Production Environments

MongoDB Atlas has a strict separation between production and non-production environments. Production and Customer data is never utilized for non-production purposes. Non-production environments are utilized for development, testing, and staging.

MongoDB Policies require the principle of least privilege and separation of duties. As a result, developers are provided access to developer environments only and production environments are limited to personnel who have an operational need and appropriate authorizations.

Firewalls and Bastion Hosts

MongoDB Atlas infrastructure is only accessible via bastion hosts. Bastion hosts are configured to require SSH keys (not passwords). Bastion hosts also require multi-factor authentication, and users must additionally be approved by senior management for backend access.

Logging and Alerting

MongoDB maintains a centralized log management system for the collection, storage, and analysis of log data for production environments. This information is used for health monitoring, troubleshooting, and security purposes. Alerts are configured on systems in order to notify SREs of any operational concerns.

Log Retention

It is the policy of MongoDB to retain its logs within its own infrastructure based on an Atlas Log Retention schedule. When the retention period is complete, logs may be destroyed. Except as otherwise indicated, logs shall be retained for the number of months or years indicated.

MongoDB is to maintain complete, accurate, and high-quality logs in storage for the duration of the time periods provided in this document. The head of Atlas engineering is responsible for authorizing, overseeing, and ensuring that logs are maintained pursuant to this document.

No logs will be destroyed if they are relevant to a pending or threatened investigation of any matter within the jurisdiction of a federal department or agency, or any other official investigation.

Retention Schedule (minimum life)	Log Source
Six years	• Web Tier • Backup Tier • Splunk Audit/Query • AWS CloudTrail • OS /var/log/secure • DB events collection audit history
One year	• UI app • Backup app • Restore app • Backup service app
One month	• Customer’s MongoDB (mongod) and audit logs • Server Automation Agent • Server Backup Agent • Server Monitoring Agent • Data “mirror” app

It is a crime for anyone to knowingly destroy logs with the intent to obstruct the proper administration of any investigation or proceeding under the jurisdiction of a federal department or agency. No logs will be destroyed if they are relevant to pending or threatened litigation matters when MongoDB is a party in the case or expected to become a party or when MongoDB has received a subpoena.

Online Archive

Online Archive is MongoDB managed object storage enabled to move your infrequently accessed data from your Atlas cluster to a MongoDB-managed cloud object storage where data can be accessed through a read-only Federated Database Instance. Access and storage to Online Archive are managed in a MongoDB-controlled instance to ensure that data is encrypted and stored securely.

Secure Deletion of Data

If a customer terminates an Atlas cluster, the following happens: it will become unavailable immediately; MongoDB, Inc. may retain a copy of the data for up to 5 days; the backup associated with the managed cluster is also terminated. If a customer terminates the backup, all snapshots become unavailable immediately. It may take up to 24 hours for all copies of the data to be deleted.

Input Validation

Input validation is done for data submitted to web applications, and verified with manual source code checks and peer reviews, as well as internal and external security team tests. Fuzz testing is also used for core product assessments.

Protection from Ransomware and Malware Attacks

One of the major concerns for enterprises today is the risk of data breaches and unauthorized exposure from ransomware. Primary vectors for malware/ransomware include malicious email, Windows AD networks, and compromised desktop browsers via infected websites. There are lot of mitigation strategies MongoDB security features offer. First, there is a true end-to-end encryption – sensitive data is protected as the data remains encrypted from the client, during transport, while at rest in the database, and while being processed in memory. With elevated features like Client Side Field Level Encryption, there is never any cleartext available in the database for sensitive workloads, even to the highest privileged administrators or sysadmins or cloud infrastructure staff and even if the database were to somehow get compromised by improperly secured credentials or some other exploit, hardened encryption technologies in use ensures that there is no data for an attacker to dump.

As part of the disaster recovery, by default MongoDB Atlas offer an option to enable backups and customers can take backups as frequently as needed. In addition, by design all Atlas clusters are highly available, multi-node replica sets spanning multiple VMs, distributed regionally, globally, or even across multiple cloud providers. These backups can be targeted to multiple media target destinations and pulled to customer remote storage via automation or manually in the Atlas console at any time to authorized users.

MongoDB Atlas offers additional safeguards depending on your business requirements and concerns.

Customers can enable Termination Protection for clusters in Atlas, to ensure the prevention of accidental deletion of your production clusters and irretrievable loss of data by enabling cluster termination protection.

- Customers can also protect all of their backups as well. The Backup Compliance Policy enables organizations to further secure business-critical data by preventing all snapshots and oplogs stored in Atlas from being modified or deleted for a predefined retention period by any user, regardless of Atlas role, guaranteeing that backups are fully WORM compliant.
- With test failover in Atlas, you will be able to test the failure of a single node up to a regional failure at the click of a button to ensure you're ready for a real-life disaster event. Testing and ensuring your cluster's resiliency is working as you expect is no longer a one-time-a-year test but now just like your CI/CD process where you can continuously test your disaster recovery throughout the year at any time.

MongoDB Personnel Access to MongoDB Atlas Clusters.

Privileged user access

As a general matter, MongoDB personnel do not have authorization to access your MongoDB Atlas Clusters. Only a small group of Privileged Users are authorized to access your MongoDB Atlas Clusters in rare cases where required to investigate and restore critical services. MongoDB adheres to the principle of "least privilege" with respect to those Privileged Users, and any access is limited to the minimum time and extent necessary to repair the critical issue. Privileged Users may only access your MongoDB Atlas Clusters via a gated process that uses a bastion host, requires MFA both to log in to our MongoDB Systems and to establish a Secure Shell connection (SSH) via the bastion host, and requires approval by MongoDB senior management.

Restricting MongoDB personnel access

MongoDB Atlas provides you with the option to entirely restrict access by all MongoDB personnel, including Privileged Users, to your MongoDB Atlas Clusters. If you choose to restrict such access and MongoDB determines that access is necessary to resolve a particular support issue, MongoDB must first request your permission and you may then decide whether to temporarily restore Privileged User access for up to 24 hours. You can revoke the temporary 24-hour access grant at any time. Enabling this restriction may result in increased time for the response and resolution of support issues and, as a result, may negatively impact the availability of your MongoDB Atlas Clusters. If you enable client-side field level encryption, even Privileged Users will be unable to access Customer Data within your MongoDB Atlas Clusters in the clear unless you provide MongoDB with the encryption keys.

Credential requirements

Privileged User accounts may only be used for privileged activities, and Privileged Users must use a separate account to perform non-privileged activities. Privileged User accounts may not use shared credentials. The password requirements described in Section 4.3.3 also apply to Privileged User accounts.

Access review and auditing

MongoDB reviews Privileged User access authorization on a quarterly basis. Additionally, we revoke a Privileged User's access when it is no longer needed, including within 24 hours of that Privileged User changing roles or leaving the company. We also log any access by MongoDB personnel to your MongoDB Atlas Clusters. Audit logs are retained for at least six years, and include a timestamp, actor, action, and output. MongoDB utilizes a combination of automated and human reviews to scan those audit logs.

Dedicated Information Security Program

Security Program

MongoDB maintains a comprehensive written Information Security Program to establish effective administrative, technical, and physical safeguards for Customer Data, and to identify, detect, protect against, respond to, and recover from security incidents. MongoDB's Information Security Program complies with applicable Data Protection Laws and is aligned with the NIST Cyber Security Framework (NIST). Additionally, MongoDB Atlas is certified against ISO 27001:2013, ISO 27017:2015, ISO 27018:2019, SOC 2 Type II, Payment Card Industry Data Security Standard v.3.2.1, and Cloud Security Alliance (CSA) Security, Trust, Assurance, and Risk (STAR) Level 2. MongoDB Atlas has also undergone a HIPAA examination validated by a qualified third-party assessor and can be configured to build HIPAA compliant applications.

MongoDB employees are required to take and attest to periodic security training. Additionally, the Security Team employs a number of education outreach efforts, such as internal security reading groups, Capture-the-Flag / Hacking Contests to teach developers security issues, hackathons, and more. Internal policies include data classification and handling and specific information regarding handling customer data.

MongoDB has a vulnerability enumeration and management program; this program identifies internet-accessible company assets, scans for known vulnerabilities, evaluates risk, and tracks issue remediation. Vulnerability scans occur at least daily, with results reporting to a centralized security dashboard. A central company-wide ticketing system is used to track all security issues until remediation.

Human Resources performs multi-residence criminal background checks on all prospective employees. The HR employee off-boarding processes includes verification of account access termination.

Application Security

MongoDB Atlas undergoes regular reviews from both internal and external security teams. Internally, MongoDB Atlas undergoes periodic risk assessments, including technical vulnerability discovery and business risks and concerns.

Additionally, the MongoDB Security Team is routinely involved in source code review, architecture review, code commit/peer review, and in security decision-making.

Application-level security testing uses a standard application assessment methodology (e.g., OWASP). Additionally, external engagements with security consults include social engineering and phishing testing. A summary of our most recent third-party penetration test is available for customers to review. Systems are patched as needed; security-related patches are applied commensurate to their severity.

Security Best Practices for Software Development

MongoDB product security teams work collaboratively on security initiatives in the SDLC. Team members are tasked with finding and preventing security issues in our products. Their responsibilities include building new security features, reviewing source code, tracking and remediating security issues, and engaging with third parties for security reviews. All customer-facing software is in a continuous integration/delivery (CI/CD) pipeline and subjected to a peer-review process. We perform hundreds of hours of automated testing to ensure correctness on every source code commit. When code commit triggers (or "hooks") are called, unit tests and library integration links are automatically run, with a pass/fail log and real-time CI dashboard update. For more information on how we follow security best practices refer to the whitepaper.

Communication and Notifications

MongoDB has an established Incident Response and Critical Communications Policy and associated processes. In the event that a security alert/event, or other signal results in MongoDB declaring a security incident, MongoDB will follow its internal incident response protocols and inform affected customers as soon as practicable. If your organization has very specific breach notification or communications requirements, please contact us directly.

Patching and Change Management

Patching of operating systems and applications are performed on a need-to-update basis. MongoDB, Inc. employees utilize automated tooling in conjunction with monitoring security bulletins for relevant software and implement patches if security issues are discovered. The MongoDB server software itself is continuously updated as new versions are released.

With respect to change management, development tasks are defined as issues for specific target releases. A release is deployed to production after it has transitioned through the requisite checkpoints, including testing, staged deployment, and management review. All internal release notes include a QA test plan.

Resources

We are MongoDB, database experts with over 43K+ customers relying on our commercial and cloud products/services. For more information, please visit mongodb.com or contact us at sales@mongodb.com.

Atlas Documentation

MongoDB Atlas download

Case studies

MongoDB Resource center

MongoDB University (Free online training)

MongoDB App Services

MongoDB Atlas database as a service

MongoDB Trust Center

MongoDB Security Hub

MongoDB Data Encryption

Technical and Security Control Measures

MongoDB Atlas for Government

FedRAMP Moderate Authorization (MongoDB Atlas for Government)

Criminal Justice Information Solutions

Cloud Shared Responsibility Model (Datasheet, Whitepaper)

© October 2024 MongoDB, Inc. All rights reserved.