MongoDB Atlas Security

February 2022
Table of Contents

Introduction

Shared Responsibility Model

Data Storage and Access Controls

Networking
   Connectivity
   IP Access Lists
   Network Peering
   Private Endpoints
   AWS VPC Topology
   Google Cloud VPC Topology
   Microsoft Azure VNET Topology

Encryption In-Transit and At-Rest

Client-Side Field Level Encryption

Customer Security Controls
   Database Authentication
   Authorization and User Rights Management
   Multi-factor Authentication
   x509 Authentication
   AWS IAM Authentication
   LDAPS Integration
   API Access
   Auditing

Infrastructure and Application Security
   Configuration Management
   Separation of Environments
   Log Retention
   Secure deletion of data

Business Continuity and Disaster Recovery

Compliance & Trust

Information Security Program Overview
Introduction

Building customer trust is a top priority at MongoDB. We understand the responsibility that 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, as well as provide our customers 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 Realm) to help our customers build to new heights securely.
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 access from the Internet by default. Additionally, MongoDB provides automated patching of the underlying infrastructure and MongoDB clusters with zero downtime.

Customers are responsible for creating users and roles to access Atlas, 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 client-side field-level encryption.

**Primer on MongoDB Atlas Organizations and Projects**

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 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.
- Maintain separate cluster security configurations. For example:
  - Create/manage different sets of MongoDB user credentials for each project.
  - Isolate networks in different VPCs.
- Create different alert settings. For example, configure alerts for Production environments differently than Development environments.
- Associate different users or teams with different environments, or give different permissions to users in different environments.
Data Storage and Access Controls

DATA CENTERS AND PHYSICAL STORAGE

MongoDB Atlas is built atop of Amazon Web Services (AWS), Microsoft Azure, and Google Cloud.

Amazon Web Services

Customer data is stored in MongoDB Atlas systems; these systems are single-tenant dedicated AWS EC2 virtual servers that are created solely for an Atlas Customer. These virtual servers are isolated within their own dedicated Virtual Private Cloud (VPC) and do not share logical data storage or processing with other customers. Amazon AWS data centers are compliant with a number of physical security and information security standards. Please visit AWS’s Compliance page if more detail regarding physical security is required.

Microsoft Azure

Customer data is stored in MongoDB Atlas systems; these systems are single-tenant dedicated Microsoft Azure Virtual Machines that are created solely for an Atlas Customer. These virtual servers are isolated within their own dedicated VNet and do not share logical data storage or processing with other customers. Microsoft Azure data centers are compliant with a number of physical security and information security standards. Please visit Microsoft’s Compliance website if more detail regarding physical security is required.

Google Cloud

Customer data is stored in MongoDB Atlas systems; these systems are single-tenant dedicated Google Cloud virtual machine instances that are created solely for an Atlas customer. These virtual servers are isolated within their own dedicated Virtual Private Cloud (VPC) and do not share logical data storage or processing with other customers. Google Cloud data centers are compliant with a number of physical security and information security standards. Please visit Google’s Compliance website if more detail regarding physical security is required.

Please note: MongoDB Atlas customers deploying M0, M2, and M5 instances will use a multi-tenant system.

Terminology

**Project**: Contains one or more MongoDB Atlas clusters with a shared set of authorization and network configurations. Dedicated clusters on AWS and Azure respectively leverage one VPC or VNet per region per Project. Dedicated clusters on Google Cloud leverage one global VPC per project.

**Cluster**: A MongoDB Atlas deployment item (a replica set or sharded cluster) that contains MongoDB databases to connect to.

**VPC**: Virtual Private Cloud, or virtual network, on AWS or Google Cloud.

**VNet**: Virtual network on Azure (equivalent to a VPC on AWS or Google Cloud).

**DNS**: Domain Name Service
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 636 on the customer side open to inbound traffic by Atlas

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 replica set to sharded cluster, an addition of shards, or a region change – will require new IP addresses be used.

IP Access Lists

By default, your MongoDB Atlas cluster will have no 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 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 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.
Before deploying your first cluster, you can choose the Atlas-side CIDR block via the VPC peering wizard. The Atlas VPC CIDR block must not overlap with the peer VPC CIDR block. Atlas uses the specified CIDR block for all other VPC peering connections created in the project and limits the number of MongoDB nodes per VPC based on the CIDR block and the region selected for the project. For example, a project with an Atlas VPC CIDR block of /24 is limited to the equivalent of 27 3-node replica sets.

See the following sections for details on network peering and DNS resolution for different cloud providers.

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

ATLAS WEB SERVICES 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.

Private endpoints are available on AWS via AWS PrivateLink, Azure via Azure Private Link and Google Cloud via Private Service Connect.
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 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.
Customers are able to choose the geographical region(s) for their database clusters and storage. MongoDB Atlas supports a large number of AWS, Azure, and Google Cloud regions globally; this includes US, EMEA and APAC locations.

**Encryption In-Transit and At-Rest**

TLS and authentication (SCRAM) are enabled by default and cannot be disabled. Traffic from clients to Atlas (both database clusters and the web UI / control plane) is authenticated and encrypted in-transit, and traffic between the customer’s internally managed MongoDB nodes is also authenticated and encrypted in-transit using TLS. TLS 1.2 is the default; customers can select TLS 1.1 or 1.0 if needed (but note that MongoDB 4.0 and later disables support for TLS 1.0 where TLS 1.1+ is available). The MongoDB Security Team continuously monitors the status of transport protocols, and requirements are continually updated in order to ensure weak ciphers are deprecated.

**Amazon Web Services**

Encryption-at-rest is automated using AWS’s transparent disk 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 transparent disk 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 transparent disk encryption, which uses Advanced Encryption Standard (AES) algorithm with 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 (different key than storage level). All keys are fully managed by Google Cloud.

**Encryption Key Management**

Customers running MongoDB Atlas may choose to “bring their own key” and enable database-level encryption for sensitive workloads via the WiredTiger Encrypted Storage Engine. All Atlas databases and snapshot backups use strong volume (disk) encryption by default to protect data at rest. 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.
Key Rotation

Customers who require key rotation can use KMS and set the master key rotation policy for automatic rotation. Whether KMS or some other solution is integrated into the local key service (via, e.g., Hashicorp Vault, AWS Secrets Manager, Google Cloud KMS, or Azure Vault) we recommend that customers create IAM (Identity Access Management) profiles for access to those services, and make the scope very narrow — only encrypt/decrypt or retrieve/store for a single key or secret, and then rotate the identity keys/credentials to that IAM service account profile.

Role of the Customer Master Key

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.

WARNING: The process of destroying a Master Key is typically executed after a project is complete and the database is no longer needed. After the Master Key is destroyed, all project cluster data becomes inaccessible and irrecoverable, including previously encrypted backups. If project data is needed in the future, please consider archiving options such as off-cloud backups with separate key management.
Key and Encryption Flow Overview for a New Atlas Project

1. Customers create Atlas project-specific Master Keys (and the associated delegated key manager identity (e.g., IAM) credentials to access them) by generating a new Key Vault in Azure, a new customer Master Key in AWS KMS or a new Google Cloud Service Account Key, with limited (scoped) rights.

2. Atlas uses the identity credentials (key ID and secret ID) for the Master Key on the customer’s behalf to request that the key service generate data encryption keys, fetch encrypted versions of those keys, and decrypt them on demand. Plaintext key material for the Master Key itself never leaves the boundary of the key management service, including the backing keys stored in highly hardened FIPS 140-2 Level 2 Hardware Security Modules and neither the customer nor Atlas support engineers are able to access Master Keys.

3. Atlas never stores its data keys unencrypted. Plaintext data decryption keys are only present in isolated memory buffers on running database instances, and are never persisted or paged to disk.

4. Atlas only requires a single delegated customer Master Key with IAM permissions limited to Encrypt and Decrypt API operations. This model ensures that customers are only granting narrowly scoped access to start and manage their project clusters with encryption at rest.

5. Atlas Web UI / control plane encryption keys also would be required to decrypt the credentials needed to encrypt and decrypt with the Master Key. In the event of a breach of Master Key KMS, Vault, or Cloud KMS credentials, if a customer were to revoke key management access from Atlas, any exposed credentials would be rendered useless and the underlying cluster data would remain unreadable.

6. The Master Key ID, IAM ID, and secret key are encrypted and transported securely from Atlas configuration to a highly segmented management control plane, restricted from MongoDB staff. In addition, the key material of the Master Key never leaves the key management service (protected by a Hardware Security Module or HSM), and cannot be retrieved by anyone, including MongoDB engineers. Atlas customers may revoke MongoDB access to their key manager at any time, to prevent decryption of data.

7. Users can generate a unique Master Key for each Atlas project should they choose.

8. Atlas uses the delegated Master Key to encrypt and decrypt MongoDB master keys, which are used to encrypt database keys.

9. Within a given replica set cluster, each MongoDB database (namespace for collections) will generate a unique per-database key. As with all keys, these are never stored plaintext on disk, but wrapped via envelope encryption with the data master key, derived from the customer’s Master Key.

10. Atlas uses the same delegated IAM user credentials and Master Key settings for all clusters in a project for which Encryption Key Management is enabled.

11. Master Key rotation is a customer’s responsibility and can be performed at any time. Atlas automatically rotates its data master keys every 90 days.

WARNING: Modifying, deleting, or disabling the Master Key (Customer Master Key in AWS, Key Vault key in Azure, or Google Cloud Service Account Key in Google Cloud) or key management provider credentials used by Atlas will make a cluster inaccessible and irrecoverable. Doing so will block the database service from performing the requests to the key management provider for unwrapping the keys needed for cluster encryption/decryption operations, and any cluster restart or unpause commands will result in an error as the cluster nodes will no longer successfully launch.
Client-Side Field Level Encryption (FLE) 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 is instead handled exclusively within the MongoDB drivers in the client, before sensitive data leaves the application.

As a result, all encrypted fields sent to the MongoDB server – whether they are resident in memory, in system logs, at-rest in storage, and in backups – are rendered as ciphertext. Neither the server nor any administrators managing the database nor cloud infrastructure staff have access to the encryption keys. Unless the attacker has a compromised DBA password, privileged network access, AND a stolen client encryption key, the data remains protected, securing it against sophisticated exploits.

MongoDB's Client-Side FLE complements 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
- Compromising your ability to query encrypted data
- Significantly impacting database performance

By securing data with Client-Side FLE 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.

You can also more easily comply with “right to erasure” mandates in modern privacy legislation such as the GDPR and the CCPA. When a user invokes their right to erasure, you simply destroy the associated field encryption key and the user’s Personally Identifiable Information (PII) is rendered unreadable and irrecoverable to anyone.

Client-Side FLE Implementation

FLE 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.

Client-Side FLE uses standard NIST FIPS-certified encryption primitives including AES at the 256-bit security level, in authenticated CBC mode: AEAD AES-256-CBC encryption algorithm with HMAC-SHA-512 MAC.

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). Client-Side FLE integrates with 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.

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 1.
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, Client-Side FLE 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 Client-Side FLE, 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.

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, 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, Client-Side FLE 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 Client-Side FLE, 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.
MONGODB, INC. EMPLOYEE ACCESS

The always-on automation model of Atlas, leveraging strict change management and code review processes, means human operator intervention is authorized only by senior management and in rare cases of critical system maintenance where customer availability requires it. Formal policies and procedures have been established by MongoDB to delineate the minimum standards for logical access to platform and infrastructure hosts.

MongoDB uses a combination of technical and logical controls to limit and audit the personnel who access systems with sensitive data. Technical role-based access controls (RBAC) are in place to ensure only the set of MongoDB employees with pre-approved operational roles are granted access to MongoDB Atlas underlying systems.

Access to underlying hosts requires multi-factor authentication and a bastion host. Operational personnel permissions and entitlements are audited on a periodic basis. MongoDB engineers will never access customer data, or the systems that store and process customer data, under normal circumstances. In "break glass" reliability situations, customer data can be accessed by appropriate personnel to investigate and restore critical services.

Customers can opt into a control setting that applies to their Atlas Org, which disables MongoDB Production Support / Technical Services engineers' ability to SSH into the backend of customer infrastructure. When leveraging this setting, in the context of a support case, if Atlas Support personnel believe that SSH access will help them to provide the help our customer requires, they will ask the customer to enable
<table>
<thead>
<tr>
<th>Data Flow</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1         | Consumer information is collected and processed by MongoDB customers. This may include PII, PHI, and information subject to PCI or Federal regulations.  
NOTE: MongoDB has no visibility/control over what type of information its customers choose to store in their MongoDB clusters |
| 2         | Customers transmit consumer data via their web or application tier(s) to their Atlas database cluster. |
| 3         | The customer Atlas clusters are hosted in IaaS MongoDB instances (i.e., AWS, Google Cloud, Azure) and supported by MongoDB provided tools (i.e., Atlas DBMS, MDB agents) |
| 4         | Customers set up and update their Atlas database clusters via the Atlas web UI/Atlas backend.  
- Customer provisioning: Includes commercial data, no consumer data  
- Performance advisor: Slow query log lines directed to the Atlas backend for analysis and index suggestions. Output would also be displayed via the web UI. Data at rest, logs would be stored for 7 days in MongoDB backend and databases.  
- Real time performance panel: Allows visibility to metadata, including queries that may contain consumer data contained in customer queries – data in transit only.  
- Data explorer: customer initiating a query against their MongoDB cluster. The query results would be displayed in the web UI and may contain consumer data (depending on the customer query) – data in transit only.  
NOTE: Customers have the option to turn these features off |
| 5         | The Atlas web UI and Atlas backend are hosted in a dedicated MongoDB AWS instance |
| 6         | MongoDB access to customer data (commercial) which may potentially include access to consumer data may occur in one of the following ways:  
- .1. Case management (Salesforce): A customer choosing to upload (a sample of) data or logs containing their data to Salesforce as part of an incident  
- .2. Metadata including logs which may contain queries on customer databases, which may contain consumer data. Most of the data would not be PII, but logs may contain very limited PII as part of queries submitted  
NOTE: MongoDB does NOT access consumer data as a business as usual (BAU) activity. The above scenarios are to be treated as incidents; consumer data is removed from MongoDB infrastructure. |
| 7         | MongoDB support access covers the following scenarios:  
- .1. SSH access to customer database clusters: Access only allowed via Bastion host, hosted in Atlas backend. User would leverage MFA (Google Auth) to establish an SSH connection to the bastion host and use another SSH connection to the customer database cluster. This access is audited and restricted to a very small group of dedicated engineers (tier 2 and tier 3 support). Tier 2/3 staff go through background checks and training and have clearly defined tasks and responsibilities.  
- .2. Tier 2: Only has access to system files (restrictions applied on an infrastructure (Linux) level)  
- .3. Tier 3: Could have access to customer files (potentially containing consumer data) due to the fact they would normally have root access on the infrastructure level |
| 8         | Use of encryption for stored data, as well as backup and archiving options utilized are defined by the customers. |
Customer Security Controls

DATABASE AUTHENTICATION, AUTHORIZATION, AND USER RIGHTS MANAGEMENT

For MongoDB Atlas, we will discuss two components:

- MongoDB Atlas Web UI / Control Plane
- MongoDB Atlas Database Cluster

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 Web UI / Control Plane supports authentication via username/password and multi-factor authentication. Control plane user identities are managed in a MongoDB-controlled Okta instance, encrypted and stored securely. Federated identity with SAML identity providers such as Okta or OneLogin is supported. Users may also create and login to an Atlas control plane account using a Google Account. Authentication to the Atlas Web 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 via SCRAM 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 which 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 which 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).

LDAPS 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.

Note: Personal API keys are deprecated.

HASHICORP VAULT INTEGRATION

You can use HashiCorp Vault to generate and manage secrets for MongoDB Atlas database users and programmatic APIs, 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 log is 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 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.
Vault Network Diagram, AWS Clusters

*AWS PrivateLink shown for illustrative purposes, but VPC peering or public IP over TLS could also be used instead.

Vault Network Diagram, Non-AWS Clusters

*A generic, peered connection is shown for illustrative purposes, but VPC peering on GCP or Azure or public IP over TLS could be used instead.
Infrastructure and Application Security

CONFIGURATION MANAGEMENT

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 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 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 for 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.

<table>
<thead>
<tr>
<th>Retention schedule (minimum life)</th>
<th>Log source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Six years</td>
<td>Web Tier</td>
</tr>
<tr>
<td></td>
<td>Backup Tier</td>
</tr>
<tr>
<td></td>
<td>Splunk Audit/Query</td>
</tr>
<tr>
<td></td>
<td>AWS CloudTrail</td>
</tr>
<tr>
<td></td>
<td>OS /var/log/secure</td>
</tr>
<tr>
<td></td>
<td>DB events collection audit history</td>
</tr>
<tr>
<td>One year</td>
<td>UI app</td>
</tr>
<tr>
<td></td>
<td>Backup app</td>
</tr>
<tr>
<td></td>
<td>Restore app</td>
</tr>
<tr>
<td></td>
<td>Backup service app</td>
</tr>
<tr>
<td>One month</td>
<td>Customer’s MongoDB (mongod) and audit logs</td>
</tr>
<tr>
<td></td>
<td>Server Automation Agent</td>
</tr>
<tr>
<td></td>
<td>Server Backup Agent</td>
</tr>
<tr>
<td></td>
<td>Server Monitoring Agent</td>
</tr>
<tr>
<td></td>
<td>Data “mirror” app</td>
</tr>
</tbody>
</table>
It is a crime for anyone to knowingly destroy logs with the intent to obstruct the proper administrative 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.

**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 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.

**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).

**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 cluster 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 backup schedule.

---

\(^1\) 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
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 enable 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.

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 to manage the impact and resolution.

COMPANY-WIDE EXECUTIVE REVIEW

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 for the following where it pertains to usage of the Atlas product:

- MongoDB Drivers
- MongoDB Connectors, including BI and Spark
- Authentication to the Atlas cluster
- AWS, Azure, and Google Cloud related questions
- Performance
- Data Migrations

This document applies exclusively to Atlas support and any of the list above that integrate with Atlas.
**SUPPORT CHANNELS**

**Atlas Support Chat**

The Atlas product includes chat support - this channel is staffed with specially trained triage personnel who assess customer questions and route them to the appropriate team within MongoDB for resolution. Technical issues will typically be routed to the Technical Services Engineering organization; sales questions will be routed to Sales, etc. The staff of this channel are equipped to answer basic technical questions about the product and its use, including light troubleshooting of connection issues (network, security) and onboarding (building a cluster, pricing, adding users, etc.).

**MongoDB Support Portal**

Customers who have purchased an Atlas support plan will receive an account for the MongoDB Support Portal. These customers can access the MongoDB Support Portal with their Atlas username and password. The Support Portal allows customers to create cases to engage the Technical Services team, view the status of their existing cases, and search the customer-only Knowledge Base.

Atlas for Government has a separate support portal — support.mongodbgov.com — to engage with MongoDB Support.

When logged into the Atlas UI, customers can also open a support ticket via the Request Support button. All further interactions will occur in the Support Portal.

**Phone**

For urgent issues or off-hours issues, customers with an Atlas Pro subscription or higher can reach the Technical Services Engineering (TSE) team via phone; the numbers are listed in the MongoDB Support Portal. These numbers are staffed 24x7x365 for high priority issues. During these calls, customers should provide their name, company, and contact details. If possible, customers should file a support case prior to calling, so that engineers have all the information needed to proceed on the case. It is permissible to escalate an existing case via a change in priority either via phone call or via clicking on the escalate button in the case view.

**CUSTOMER ON-BOARDING**

After the purchase of an Atlas support plan, customers will receive directions on how to access the support portal, how to create a support case, and contact information for Technical Services Engineering. Customers are also given instructions for how to add and enable more users for their account, so that multiple persons within their organization can access support.

**SUPPORT CONSIDERATIONS**

Customers should establish both a single owner and one or more admin contacts for the account. Only these contacts will be able to add additional users to their Salesforce group. If desired, customers can create a generic contact for their company in the event that we need to reach your entire team via adding an email alias.

Filing cases in the Support Portal or via Atlas UI: The global MongoDB Technical Services Engineering (TSE) team monitors all cases filed in the Support Portal and via the Atlas UI. Other channels (e.g., bug reports, product feedback) are not monitored actively by TSEs, do not trigger SLAs, and do not show up on global TSE reports and dashboards.

**CASE SEVERITY AND DEFINITIONS**

Customers must select the severity of their technical issue or question when opening a support case with MongoDB. Selection of severity by the customer will affect SLA response times. More information on our Support Policy and Severity handling is available here.
PRODUCT AND ENGINEERING FEEDBACK

Customers can submit product feedback (feature requests, improvements, or bug reports) at any time in various public facing JIRA projects for the appropriate product. Otherwise, the TSE assigned to the customer support case can also take that feedback and file the appropriate requests. Depending on urgency and situation, this may result in immediate production releases, backport to previous versions of the product, and/or prioritization for upcoming releases. If the TSE is informed of this information (the fix version and release plan) they will communicate that clearly to the customer.

ATLAS TECHNICAL ISSUE MANAGEMENT

As a fully managed and hosted solution, the Atlas product has some differences from standard support, both in terms of business process and in terms of engineering infrastructure.

Cloud Infrastructure

The Atlas system will automatically detect and repair most problems without human intervention. If a MongoDB process crashes, it will automatically be restarted. If the underlying virtual machine is unhealthy, it will automatically be replaced. The Atlas system also actively monitors for any issues that could not be successfully repaired and surfaces them on the admin dashboard of Cloud Operations Engineers (COE). If scaling is required, a COE will create a customer facing support ticket and make this recommendation. The uptime SLA for Atlas is publicly available here.

Cloud Operations Engineers

Upon receiving an alert or observing any Atlas cluster healing issues, the COE for Atlas will take remediation steps to diagnose and repair the issue. Examples of situations where this might occur include:

- When a MongoDB process is down
- When a MongoDB replica set has an unhealthy member

A COE will access the host and take corrective action whether that be restarting mongod, reclaiming storage, other system administration tasks, or replacing the instance. If reaching out to the customer is required, the COE will do this in a customer visible support ticket.

Cloud Support Engineers

If the customer has noticed an issue or has questions regarding their application or cluster behavior and would prefer not to utilize the chat support feature that is included with Atlas, they can file a support ticket here and a Technical Services Engineer will respond to their inquiry within the defined SLA support window.

ESCALATIONS

Technical issues are escalated through the Technical Support team based on internal procedures of MongoDB. Customers can escalate a case most quickly by clicking on the Escalate button at the top of each case; this will automatically engage the proper parties and send alerts to the team. If for some reason clicking the button is not an option, calling the Support phone lines is the next best option.

If an issue cannot be resolved by the TSE, it follows an internal escalation procedure. Depending on the nature and severity, an issue routes to one or more of the following teams:

- Site Reliability Engineering (SRE): responsible for infrastructure uptime and involved in any system-wide issues with the infrastructure
- Product Engineering: responsible for building products (Atlas, Realm, Compass, etc.) and involved in issues of product functionality, workarounds and/or fixes. Features and issues that do not require immediate resolution are scheduled with Product Management.
• Server Team: Product Engineering team responsible for MongoDB database product
• Product Security: responsible for security and compliance status and features of products
• Corporate Security: responsible for security and compliance of infrastructure and services
• Senior Executive Management

Compliance & Trust


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 establish 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 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 →

FEDRAMP

FedRAMP is a US Federal Government program that promotes the adoption of secure cloud services across the government by providing a standardized approach to security and risk assessment for cloud technologies and federal agencies. MongoDB Atlas for Government is FedRAMP Ready and FedRamp In-Process.
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

Learn More →
Information Security Program Overview

SECURITY PROGRAM

MongoDB Inc. has a dedicated Information Security Team. This team is responsible for the Information Security program at MongoDB. MongoDB internal security practices and policies are aligned to be compliant with ISO 27002 controls. Program priority is regularly reviewed, and is based upon threat modeling and internal continual capability and maturity assessments.

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, which includes technical vulnerability discovery as well as 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 includes 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.

COMMUNICATIONS 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 the MongoDB experts with over 31K+ customers relying our commercial products. We offer software and services to make your life easier. For more information, please visit mongodb.com or contact us at sales@mongodb.com.

- Case studies (mongodb.com/customers)
- Resource center (mongodb.com/resource-center)
- Free online training (university.mongodb.com)
- Documentation (docs.mongodb.com)
- MongoDB Realm (mongodb.com/realm)
- MongoDB download (mongodb.com/download)
- MongoDB Atlas database as a service (mongodb.com/atlas)
- MongoDB Atlas for Government (mongodb.com/cloud/atlas/government)