Introduction

Security is a top priority at MongoDB and we understand how important your data is to you and those who depend on you. MongoDB has been entrusted with a significant variety and amount of sensitive application and user data. We do not take our responsibility lightly; we work diligently to continuously improve security processes and controls, as well as provide our customers the right features to secure data as necessary.

MongoDB handles data with the utmost care and integrity, designing our systems to reduce the chance of errors from human factors, employing industry standard information security best practices, and continuously testing to find and fix vulnerabilities. Whether it’s encrypting your data from end to end, or providing you with important access control features, we want customers to have confidence in the systems and services handling sensitive workloads as they are transported, processed, and stored.

We believe transparency in security processes and controls is essential. We frequently get asked to document our security controls and standards for MongoDB Atlas. For users of Atlas, it is imperative to know who can access data when, and what controls prevent unauthorized access. These are the same concerns that MongoDB Inc. has with any of our vendors. We’re hopeful this document provides clarity by offering you a detailed understanding of MongoDB Atlas security controls and features as well as a deeper view into the cloud automation behind the scenes.

WHAT IS MONGODB ATLAS?

MongoDB Atlas is a database-as-a-service (DBaaS) created by the experts who design and engineer MongoDB. With Atlas, MongoDB Inc. helps customers by managing the underlying systems, operations, and infrastructure components that make up a database deployment, allowing customers to focus on their applications instead of database systems. This document is focused on Atlas security features; if you would like more information about the database as a service overall, please see our FAQ.

Primer on 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.
- Associate different users or teams with different environments, or give different permissions to users in different environments.
- 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.
- Deploy into different regions or cloud platforms.
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 Platform (GCP).

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 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.

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

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 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.

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

Google Cloud Platform

Customer data is stored in MongoDB Atlas Systems; these systems are single-tenant dedicated GCP virtual machine instances that are created solely for an Atlas Customer. These virtual servers are isolated within their own Virtual Private Cloud (VPC) and do not share logical data storage or processing with other customers.

Google Cloud Platform 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 instance sizes will use a multi-tenant system.

NETWORK ISOLATION

Terminology

Network ports: MongoDB requires the following network ports for Atlas

- 27016 for shards
- 27015 for the BI connector
- 27017 for server
- The network ports cannot be changed and TLS cannot be disabled.
- Can be isolated by IP whitelist
- If LDAP is enabled, MongoDB requires LDAP network 636 on the customer side open to 0.0.0.0 (entire Internet) traffic.

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 GCP leverage one global VPC.

Cluster: A MongoDB Atlas deployment item that contains MongoDB databases to connect to.

VPC: Virtual Private Cloud, or virtual network, on AWS or GCP.
VNet: Virtual network on Azure (equivalent to a VPC on AWS or GCP).

DNS: Domain Name Service

DNS split-horizon: The same name leads either to a private IP address or a public one, depending on the client sending the query; local client machines can access the service through the local network, without the need to use public networks to reach the service.

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 IP whitelisting 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 whitelist in Atlas.

MongoDB Atlas on Azure cluster public IPs are reused when cluster are scaled, but can change as of Feb 2018 when individual nodes are healed. Customers have leveraged DNS lookups to refresh cluster IPs after healing events, but in order to reduce the burden, MongoDB Atlas will move to leverage static public IPs for clusters in the near future. This means that a replica set of sharded cluster’s current public IPs will be static upon healing and vertical scaling events. However, topology changes such as a conversion from replica set to sharded cluster, or addition of shards, will require new IP addresses to be used.

VNet peering is available for MongoDB Atlas deployments on Azure. Once enabled, DNS entries for clusters will resolve only to private IPs (Azure does not offer built-in split-horizon DNS; all private IPs is required), meaning that it will not be possible to connect to a cluster from outside the peered VNet. VNet peering is available for single-region MongoDB Atlas clusters (multi-region Atlas clusters on Azure will require public IP whitelisting), but it is possible to peer with a VNet in another region.
AMAZON 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).

Many of MongoDB Atlas customers have their applications living within their own AWS account and virtual private cloud. The AWS VPC Peering option allows peering your MongoDB Atlas network to your own VPC network, thereby allowing your encrypted traffic to never traverse the public internet and instead use your internal private network. Additionally, this eliminates the need to whitelist IP addresses. Cross-region VPC peering is also supported on AWS.

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.

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 in an AWS region supporting 3 availability zones and an Atlas CIDR VPC block of /24 is limited to the equivalent of 27 3-node replica sets.

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.

Before deploying your first cluster, you can choose the Atlas-side CIDR block via the VPC peering wizard. 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 whitelist.

Please note: Atlas does not need access into peered VPCs except when LDAP is enabled. In that scenario, Atlas clusters need to reach the customer’s LDAP 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.

Some customers leverage custom DNS solutions that cannot take advantage of built-in split horizon DNS. For customers in this situation today, public IP whitelisting can be leveraged in lieu of VPC peering. Note that MongoDB is targeting introducing an option for users of AWS VPC peering that will make it so that DNS entries for clusters will resolve only to private IPs. This capability is anticipated to be available in the future. MongoDB Atlas on AWS cluster public IPs are reused when cluster are scaled, but can change as of Feb 2018 when individual nodes are healed. Customers have leveraged DNS lookups to refresh cluster IPs after healing events, but in order to reduce the burden, MongoDB Atlas will move to leverage static public IPs for clusters in the near future. This means that a replica set of sharded cluster’s current public IPs will be static upon healing and vertical scaling events. However, topology changes such as a conversion from replica set to sharded cluster, or addition of shards, will require new IP addresses be used.

Potential future options leveraging PrivateLink

PrivateLink offers two potential advantages over peering: transitive connectivity (e.g. from another peered VPC or DirectConnect) and more comprehensive control over the network trust boundary. However, an architecture mismatch between PrivateLink (primarily for delivering SaaS style applications that sit behind a network load balancer) and MongoDB Atlas (delivered as a pure infrastructure service) makes it difficult to estimate when PrivateLink capability will be available on MongoDB Atlas. Both long and short-term strategies are being explored to potentially fit Atlas into the PrivateLink framework.

GOOGLE CLOUD PLATFORM 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 Platform (GCP) VPC.

Atlas deploys a cluster in a dedicated global GCP VPC, and then uses authentication and IP whitelisting to isolate the service. A logical service in GCP 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 whitelist in Atlas. MongoDB Atlas on GCP cluster public IPs are reused when cluster are scaled, but can change as of Feb 2018 when individual nodes are healed. Customers have leveraged DNS lookups to refresh cluster IPs after healing events, but in order to reduce the burden, MongoDB Atlas will move to leverage static public IPs for clusters in the near future. This means that a replica set of sharded cluster’s current public IPs will be static upon healing and vertical scaling events. However, topology changes such as a conversion from replica set to sharded cluster, or addition of shards, will require new IP addresses be used.

VPC peering is available for MongoDB Atlas deployments on GCP. Once enabled, DNS entries for the clusters will resolve only to private IPs (GCP does not offer built-in split-horizon DNS; all private IPs is required), meaning that it will not be possible to connect to a cluster from outside the peered VPC. Because GCP VPCs are global, MongoDB Atlas will deliver VPC peering for single-region and multi-region MongoDB Atlas clusters.
REGION SELECTION

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 GCP 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 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. Administrators can control the TLS version required for their database clusters (e.g. requiring TLS 1.2), with TLS 1.1 being the default. 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 Platform

Encryption for data at rest is automated using GCP’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 GCP.

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 GCP 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.
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.

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 key manager credentials to access them) by generating a new Key Vault in Azure, a new Customer Master Key in AWS KMS or a new GCP Service Account Key, with limited (scoped) rights.

2. Atlas uses the credentials (key ID and password) 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, 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 limited 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.

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 Atlas engineering. 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 GCP Service Account Key in GCP) 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.
With MongoDB 4.2, you can now selectively encrypt individual document fields, each optionally secured with its own key and decrypted seamlessly on the client.

Our implementation of Field Level Encryption is totally separated from the database, making it transparent to the server, and instead handled exclusively within the MongoDB drivers on the client. All encrypted fields on the server – stored in-memory, in system logs, at-rest, and in backups – are rendered as ciphertext, making them unreadable to any party who does not have client access along with the keys necessary to decrypt the data. This is a different and more comprehensive approach than the column encryption used in many relational databases. As most handle encryption server-side, data is still accessible to administrators who have access to the database instance itself, even if they have no client access privileges.

In developing Field Level Encryption, we have worked with two of the world’s leading authorities on database cryptography, including a co-author of the IETF Network Working Group Draft on Authenticated AES encryption. Drawn from academia and industry, these teams have provided expert guidance on our FLE design and reviewed the FLE software implementation.

Field Level Encryption serves as an important addition to your defense-in-depth security strategy. Consider a typical MongoDB deployment from a risk management perspective:

- With filesystem encryption alone, system administrators or attackers who elevate system-level user privileges still have access to plaintext database files both on server storage and in memory.
- The MongoDB Encrypted Storage Engine provides a way to mitigate filesystem and backup file access risks by encrypting all MongoDB data before it is written to disk, and ensuring keys are non-persistent. Any attacker obtaining database files from the filesystem would be unable to read them. However, administrators and compromised authenticated database users still have access to the underlying data on a running instance.

- Using MongoDB FLE you can now protect individual fields with all key management, encryption, and decryption operations occurring exclusively outside the database server. With FLE enabled, a compromised administrator or user obtaining access to the database, the underlying filesystem, or the contents of server memory (for example, via scraping or process inspection) will only see unreadable encrypted data. While storage engine encryption and FLE can be used independently, they bring the greatest levels of protection when used together.

MongoDB Field Level Encryption provides:

- **Automatic, transparent encryption:** Application code can run unmodified for most database read and write operations when FLE is enabled. Other client-side approaches require developers to modify their query code to use the explicit encryption functions and methods in a language SDK.
- **Separation of duties:** System administrators who traditionally have access to operating systems, the database server, logs, and backups cannot read encrypted data unless explicitly given client access along with the keys necessary to decrypt the data.
- **Regulatory Compliance:** Comply with “right to be forgotten” conditions in new privacy regulations such as the GDPR – simply destroy the customer key and the associated personal data is rendered useless.
- **Minimal performance penalty:** As encryption is handled on the client, impacts to server performance are minimal when working with encrypted fields.

For isolation, each field can be encrypted with its own unique key natively integrated with external key management services backed by FIPS 140-2 validated Hardware Security Modules (HSMs) such as Amazon’s KMS.

To understand more about implementing FLE, let’s take a look at the flow of a query submitted by an authenticated client, as represented in the figure below.
1. Upon receiving the query, the MongoDB driver analyzes the query to determine if any encrypted fields are involved in the filter.

2. The driver requests the fields’ encryption keys from the external key manager.

3. The key manager returns the keys to the driver, which then encrypts the sensitive fields.

4. The driver submits the query to the MongoDB server with the encrypted fields rendered as ciphertext.

5. MongoDB returns the encrypted results of the query to the driver.

6. The query results are decrypted by the keys held by the driver, and returned to the client.

Since the database server has no access to the encryption keys, certain query operations such as sorts and range-based queries on encrypted fields are not possible unless implementing a client-side solution, which may have additional performance impacts. As a result, FLE is best applied to selectively protect just those fields containing highly sensitive, personally identifiable data such as credit card information and social security numbers.

Field Level Encryption is especially powerful when using managed database services like MongoDB Atlas. In Atlas, all cluster storage and backups are encrypted at rest by default. You can provide an additional layer of encryption by protecting the database keys using the cloud providers’ key management service. With the addition of FLE, keys are protected in an isolated, customer-managed AWS KMS account (additional key management solutions are coming online in the near future). Atlas Site Reliability Engineers and Product Engineers have no mechanism to access FLE KMS keys, rendering data unreadable to any MongoDB personnel managing the underlying database or infrastructure for you. By combining these security capabilities, you eliminate common security concerns when moving database workloads to managed services in the cloud. This is because you both control and manage the encryption keys, rather than having the database operator manage the keys for you.

At this time, driver support for FLE is at beta status, so you should not use FLE in production for now. For more information please see the MongoDB Server 4.2 documentation.
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 back-end 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 another setting, which temporarily (for 24 hours) restores Atlas Support access via SSH to a specific cluster.
<table>
<thead>
<tr>
<th>Data Flow</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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</td>
</tr>
<tr>
<td>2</td>
<td>Customers transmit consumer data via their web or application tier(s) to their Atlas database cluster.</td>
</tr>
<tr>
<td>3</td>
<td>The customer Atlas clusters are hosted in IaaS MongoDB instances (i.e. AWS, GCP, Azure) and supported by MongoDB provided tools (i.e. Atlas DBMS, MDB agents)</td>
</tr>
</tbody>
</table>
| 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) 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, however, 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. Customer federation via SAML is available in early access. Please contact your MongoDB sales contact for more information.

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.

IP ADDRESS WHITELISTING

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 whitelisting 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 whitelist for the appropriate MongoDB Atlas project.

Atlas also supports creating temporary whitelist 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.

MULTI-FACTOR AUTHENTICATION

MongoDB Atlas also allows customers to choose to optionally utilize multi-factor authentication, or require all of the users in their Atlas Organization to use multi-factor authentication. 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.

LDAP INTEGRATION

User authentication and authorization against MongoDB Atlas clusters can be managed via a customer’s Lightweight Directory Access Protocol (LDAP) server over TLS. A single LDAP configuration applies to all database clusters within an Atlas project. For customers running their LDAP 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 whitelist entry. The creation and deletion of keys will be logged in the Atlas activity feed.

Note: Personal API keys are deprecated.

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. Admins can select the actions that they want to audit, as well as the MongoDB users, Atlas roles, and LDAP 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.
Infrastructure and Application Security

**CONFIGURATION MANAGEMENT**

MongoDB Atlas’ 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 Virtual Machines on GCP use in-house built machine images with hardening applied, 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 NONPRODUCTION 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 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. 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.
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 Atlas runs on infrastructure with a high level of availability and a resilient IT architecture. MongoDB has designed its systems to tolerate system or hardware failures with minimal customer impact. Business Continuity Management at MongoDB is under the direction of the Operations Group.

Availability

MongoDB Atlas data centers are built in clusters in various global regions. All data centers are online and serving customers; no data center is "cold." These data centers are operated and maintained by MongoDB Atlas’s Infrastructure provider(s); MongoDB Atlas customers choose the infrastructure provider they wish to use, as well as what region(s) to operate in, offering flexibility to meet unique requirements.

In case of failure, automated processes move customer data traffic away from an affected area. Core applications are deployed in an N+1 configuration, so that in the event of a data center failure, there is sufficient capacity to enable traffic to be load-balanced to the remaining sites. This means that availability can be physically separated within a typical metropolitan region and be located in lower risk floodplains (specific flood zone categorization varies by Region). In addition to discrete uninterruptible power supply (UPS) and onsite backup generation facilities, each are fed via different grids from independent utilities to further reduce single points of failure. All are redundantly connected to multiple tier-1 transit providers.

MongoDB Atlas provides flexibility to place instances and store data within multiple geographic regions as well as across multiple availability zones within each region. Distributing applications across multiple availability zones provides the ability to remain resilient in the face of most failure modes, including natural disasters or system failures.

Each availability zone is designed as an independent failure zone. Every MongoDB Atlas cluster is deployed as a self-healing replica set which provides automatic failover in the event of a failure. Recovery point objectives (RPO) are met with replication and tunable durability, which acknowledges writes from one node to majority members of a replica set. Replica set members send heartbeats every two seconds and after 10 seconds mark an unresponsive member as inaccessible. Different architecture characteristics can affect time due to replication details but recovery typically can be under 30 seconds and recovery point can be under 10 seconds.

A minimum of three data nodes per replica set are automatically deployed across availability zones (AWS), fault domains (Azure), or zones (GCP) for continuous application uptime in the event of outages and routine maintenance.
INFRASTRUCTURE SERVICE RECOVERY

MongoDB Atlas creates and configures dedicated clusters on infrastructure provided by AWS, Azure and/or Google GCP. 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 Platform Compliance

CONTINUOUS BACKUPS

MongoDB Atlas backs up data, typically seconds behind an operational system. MongoDB Atlas ensures continuous backup of replica sets, consistent, cluster-wide snapshots of sharded clusters, and point-in-time recovery. This fully managed backup service uses Amazon S3 in a region nearest to customer database deployment. The distribution of backups are as follows:

- UK databases backup to UK Amazon S3
- Australia databases backup to Australia Amazon S3
- Germany databases backup to Germany Amazon S3
- United States databases backup to United States Amazon S3
- Ireland databases backup to Ireland Amazon S3
- Databases deployed in any other region backup to Ireland

Amazon S3 Backup data is protected using server-side encryption. Amazon S3 encrypts backup data at the object level as it writes it to disks in its data centers and decrypts it for you when you restore it. All keys are fully managed by AWS.

CLOUD PROVIDER SNAPSHOTS

Available for Atlas clusters deployed in Amazon Web Services, Microsoft Azure, and Google Cloud Platform, 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 multiregion 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 / GCP Service Account Key and IAM credentials are required to perform restores of backup snapshots. Cloud provider snapshots 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.

If you wish to use MongoDB Atlas Backups and have specific concerns regarding where your data must live, please contact us directly to ensure your compliance needs are met.

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

The MongoDB Technical Services Engineering team provides support for the GA releases of the following software:

- MongoDB Server
- MongoDB Cloud Manager
- MongoDB Atlas
- MongoDB Compass
- MongoDB Charts
- MongoDB Stitch
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 GCP 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.).

**Salesforce Service Cloud Portal**

Customers who have purchased an Atlas support plan will receive an account with MongoDB’s Salesforce Service Cloud Portal. These customers can access the Salesforce Service Cloud Portal with their Atlas username and password. There they can view their current open support cases or “Create Issue” via the UI.

**Phone**

For urgent issues or off-hours issues, customers with Atlas Pro or higher can reach the Technical Services Engineering (TSE) team via our answering service at 1 646-201-9247 / +44 808 281 2632. This number is staffed 24x7/365 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.

**Cloud Support JIRA Project**

When in the Atlas UI, customers can open a support ticket via a support button here. Customers should sign in with their Atlas username and password.

**CUSTOMER ONBOARDING**

After the purchase of their Atlas Pro support plan or higher, customers will receive an onboarding email with instructions on how to begin use of the Atlas service, account activation codes, directions 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 MMSSUPPORT: The global MongoDB Technical Support Engineering (TSE) team monitors all cases filed in the Support Portal and the MMSSUPPORT project in JIRA. JIRA tickets filed outside of these locations are not monitored actively, do not trigger SLAs, and do not show up on global TSE reports and dashboards.

**PRIORITIES AND DEFINITIONS**

Customers must select the priority of their technical issue or question when opening a support case with MongoDB. Selection of priority 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 Production Operations Engineers (POE). If scaling is required, a POE will create a customer facing support ticket and make this recommendation. The uptime SLA for Atlas is publicly available here.

Production Operations Engineers

Upon receiving an alert or observing any Atlas cluster healing issues, the POE 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
- When a MongoDB replica set has no primary

A POE 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 POE 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 TSE team based on internal procedures of MongoDB. Customers escalate existing support cases in the following order:

- Phone call
- Clicking “Escalate” button on the case
- Increasing priority of the case
- A written request on the ticket itself Informing TSE to escalate
- Contacting the Account Executive (AE) or Customer Success Manager (CSM)

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, Stitch, 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

**SOC**

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.

**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.

**EU-U.S. Privacy Shield**

The EU-U.S. Privacy Shield is a legal mechanism designed by the U.S. Department of Commerce and the European Commission that enables transfers of personal data from the EU to the United States.

MongoDB complies with the EU-U.S. Privacy Shield Framework as set forth by the U.S. Department of Commerce regarding the collection, use, and retention of personal information transferred from the European Union to the United States.

**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.

**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 (270xx). The best known and most widely discussed is 27001, which sets requirements for an information security management system (ISMS).

**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.
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 system 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. Over 15,000 organizations rely on 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 Stitch serverless platform (mongodb.com/stitch)
- MongoDB download (mongodb.com/download)
- MongoDB Atlas database as a service (mongodb.com/atlas)