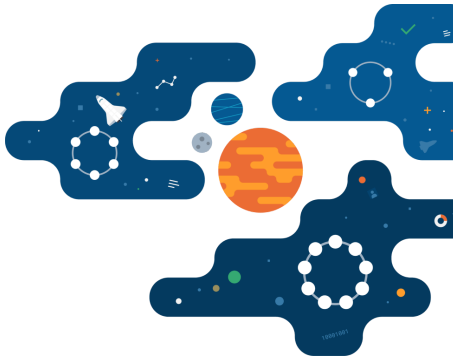
WHITEPAPER



DataStax Astra Architecture and Security Overview

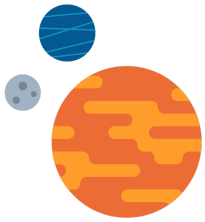
The purpose of this document is to:

- **Architecture Overview** - Provide a brief overview of the architecture of the DataStax Astra database-as-a-service platform.
- **Information Security and Privacy Program** - Describe how DataStax follows information security best practices to embed security and privacy by design including external audits and certifications.
- **Security Features and Usage** - Describe the Database security features that are included in Astra and provide guidance on how and when to use them.

DataStax

01

Architecture Overview



Astra is a cloud-native database-as-a-service built on a foundation of components of both Apache Cassandra and DataStax Enterprise (DSE) running in stateful Kubernetes clusters in multiple public clouds. Astra is compatible with open source Cassandra and includes a core subset of DSE capabilities.

Astra provides on-demand Cassandra databases on a scalable and resilient infrastructure. As a database-as-a-service offering, all Astra infrastructure -- the control plane, the Astra Cloud Console (web-based UI), and the database clusters are configured and operated within DataStax environments.

The Astra service is designed to be resilient and highly available to minimize both downtime and the need for site-reliability engineering and relies on the DataStax-built and open-sourced ***Cassandra Kubernetes operator.*** Astra is built to span multiple availability zones with multi-region support on our roadmap. Specifically, every capacity unit (CU) is composed of 3 pods running on separate availability zones.

Astra is built on multiple Cloud Native Computing Foundation (CNCF) projects: Kubernetes, Prometheus, and Envoy and uses native GKE and EKS control and management planes with built-in management sidecars, metrics collectors, and configuration builders. Additional detail is outlined in Sam Ramji's blog post entitled ***Astra: The Future of Apache Cassandra is Cloud Native.***

Core Astra Functions

Astra comprises these core functions:

- **The Core Database:** DSE, DataStax's version of Apache Cassandra optimized for performance and security management
- **Astra Data API:** Authorization and interaction with the database via REST and GraphQL
- **Secured Connection:** Endpoint for interacting directly with the database with CQL over an mTLS connection

- **CQLSH:** Integrated Cassandra Query Language Shell (CQLSH) for interactive CQL commands available directly in the Astra UI
- **DataStax Developer Studio:** Notebook style developer UI for interactive CQL commands available directly in the Astra UI
- **DataStax Bulk Loader (dsbulk):** Bulk loading tool compatible with DSE, Cassandra, and Astra; useful from migrations and fast ingests
- **DataStax Drivers:** Unified application drivers for major programming languages compatible with DSE, Cassandra, and Astra
- **Metrics dashboards:** Grafana powered metrics for Astra clusters available directly in the Astra UI.

Encryption

Users connect to Astra via a secure endpoint that provides in-transit encryption via industry-standard mutually authenticated TLS -- where both sides of the TLS connection use information in DataStax-generated certificates to verify the connection -- with unique certificates created for each database cluster.[1] Persistent storage volumes are encrypted with KMS-managed keys as are the Astra backups stored in each cloud provider's respective blob store: AWS Simple Storage Service (S3), Google Cloud Platform Cloud Storage, etc.

Architecture

The Astra control plane is built within purpose-hardened AWS accounts. Access is granted based on role and the level of access is based on need. Organization Service Control Policies (SCPs) provide guardrails, actions are logged and used for security monitoring, and the native AWS GuardDuty Intrusion Detection System[2] is enabled and monitored to protect against malicious and unauthorized access. Customer clusters are deployed in subordinate accounts and projects across public clouds. All infrastructure is written as code, which is peer-reviewed and tested in lower environments.

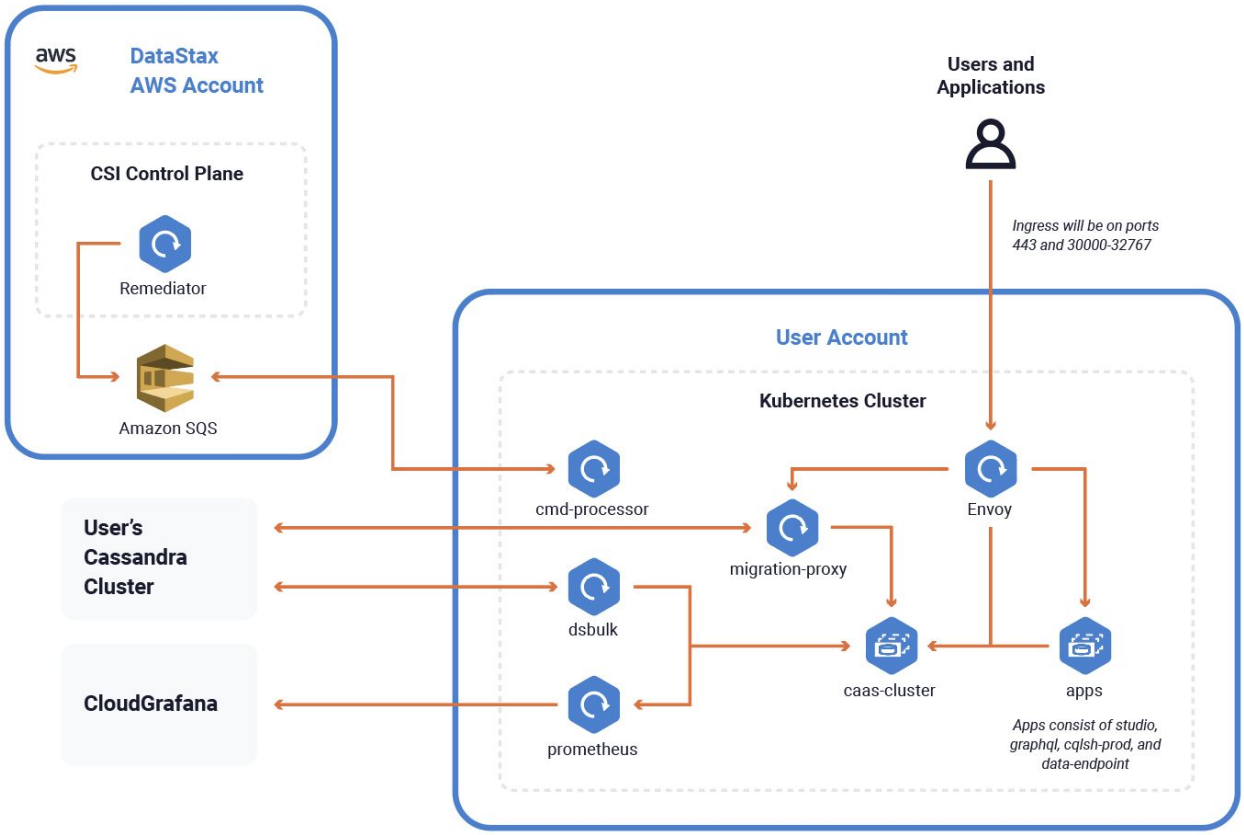
The sections below are high level illustrations of the two Astra architectures: Single-Tenant and Multi-Tenant architectures.

[1] Overview of mutual TLS: https://tyk.io/docs/basic-config-and-security/security/tls-and-ssl/mutual-tls/

[2] GuardDuty: https://aws.amazon.com/guardduty/

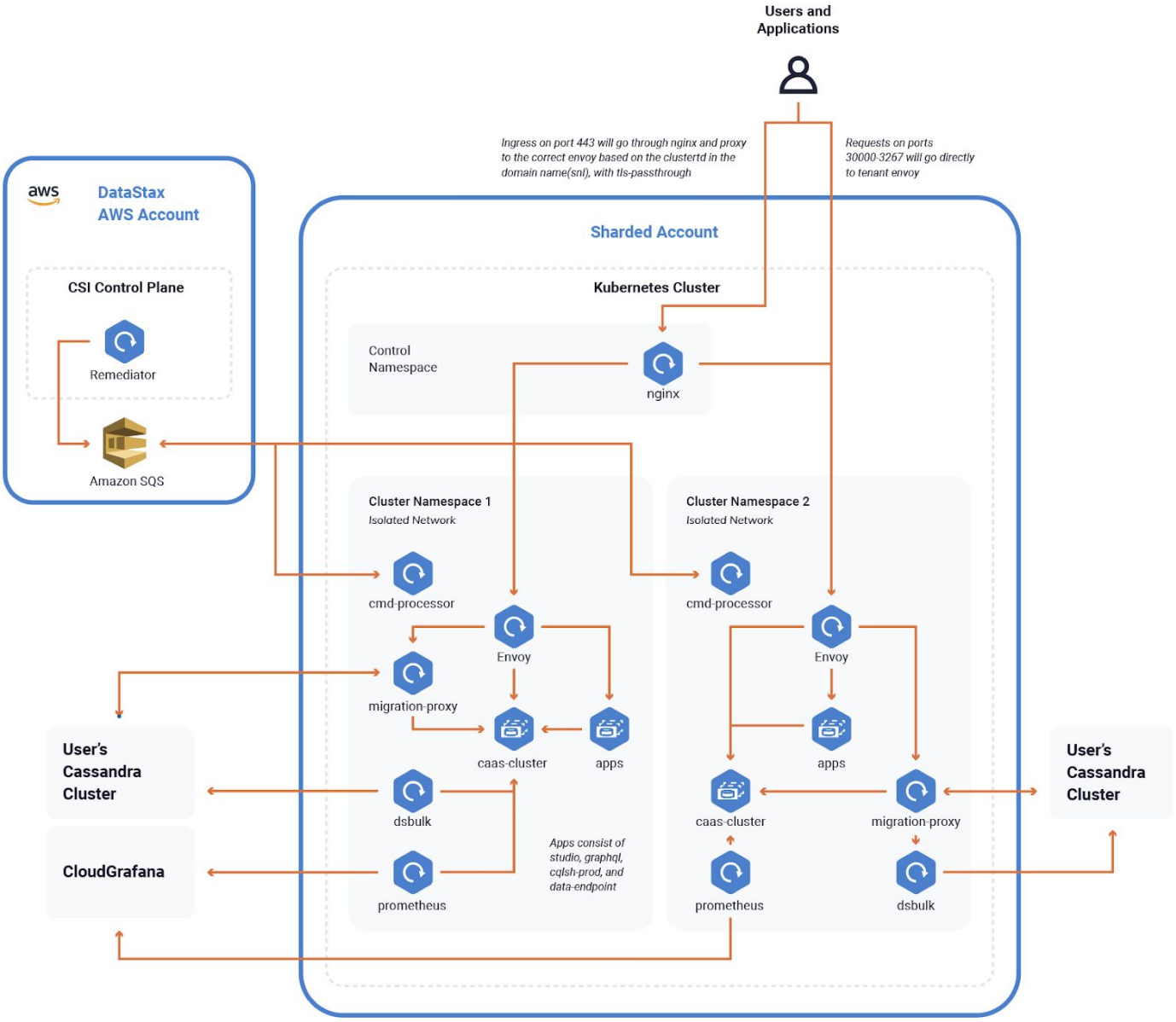
Single-Tenant

For the single-tenant version of the Astra service, each cluster is created within its own sharded sub-account inside the Astra public cloud master account: i.e., a sub-account within the AWS Organization or a Project within GCP.



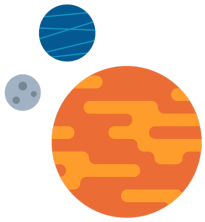
Multi-Tenant

For the multi-tenant version of the Astra service, customer resources are shared within a cluster and segregated by Kubernetes namespace and networking policies.



02

Information Security and Privacy Program



Security by Design

The DataStax security & compliance team are embedded within the Engineering organization to ensure that information security and privacy protections are built-in by design. The program is based on the NIST 800-53 CyberSecurity framework and informed by ISO 27001 best practices together with core privacy principles including those in the European Union General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA).

The Astra offering has been reviewed by external auditors which affirm that controls are in place as of May 2020 under the AICPA SOC2 framework. The Type1 report is available under NDA as of late-June 2020 and the Type2 period will end in late September 2020.

Astra-Level Protections

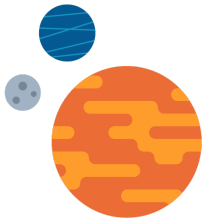
The DataStax security engineering team is responsible for building platform-level controls to support Astra confidentiality and data integrity. These include intrusion detection and prevention, custom log-based alerting, static code analysis, and container scanning.

Data is encrypted-at-rest with the use of cloud-native encryption for ephemeral instance storage, persistent backup stores, and protection of data in transit. The keys used for disk encryption for both instance storage and backup stores are stored in KMS. Astra offers the ability for customers to bring their own KMS keys to further segment data.

The security program also includes 3rd-party penetration testing at least three times per year and an ongoing bug bounty program.

03

Authentication and Access Control



User Access

Astra Cloud Portal Access

Access to the Astra cloud portal UI at ***astra.datastax.com*** is secured using a RedHat CNCF project called KeyCloak[3]. KeyCloak supports third party authentication and Astra specifically implements authentication with Google and Github for administrators and developers that need to manage databases through the UI.

DataBase Access

Developer Endpoints (REST and GraphQL)

Astra offers developer endpoints for REST and GraphQL that can be used to perform DDL and DML operations against the database. Authorization for the developer endpoints is achieved by requesting a token from the service using credentials. Tokens are active for 30 minutes after the most recent request has been made and expire after 30 minutes of inactivity.

SSL is used for all interactions with the Astra Developer endpoints. Specific instructions on how to use the developer endpoints can be found in the ***developer endpoint documentation.***

VPC Peering

Some tiers of Astra databases support VPC Peering, which allows a user's personal or corporate-owned VPC to peer with the Astra VPC where the database runs. This ensures the connection between application clients and the database is entirely over privately connected networking.

CQL

Developers, administrators and service users access the Astra database through its secure endpoint using the secure connect bundle which streamlines the process and is natively supported by the DataStax drivers. The secure connect bundle is a zip file that contains the following files:

[3] The Keycloak project documentation: https://www.keycloak.org/documentatio

```
$ unzip secure-connect-live-demo.zip
Archive:  secure-connect-live-demo.zip
  inflating: ca.crt
replace key? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: key
  inflating: cert
  inflating: identity.jks
  inflating: trustStore.jks
  inflating: config.json
  inflating: cqlshrc
  inflating: cert.pfx
```

These files include the keys and certificates required for mTLS but the user does not have to interact with them directly. It also contains the cqlshrc configuration needed to connect to astra from cqlsh. To connect to Astra from cqlsh run the following command (or use the web version of cqlsh in the Astra ui):

```
./bin/cqlsh -u username -p password -b /path/to/secure-connectdatabase_name.zip
```

At DataStax we hold that in order to ensure that security best practices are followed, systems must be secure by default and engineers must emphasize security UX when designing systems. As an example of how we follow this guideline in Astra see the comparison below between an encrypted connection against Cassandra and the secure bundle for Astra.

The traditional connection string for a cassandra database with client encryption looks as follows:

```
KeyStore ks = KeyStore.getInstance("JKS");
InputStream trustStore = new FileInputStream("client.truststore");
ks.load(trustStore, "password123".toCharArray());
TrustManagerFactory tmf =
TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
tmf.init(ks);

SslContextBuilder builder = SslContextBuilder
  .forClient()
  .sslProvider(SslProvider.OPENSSL)
  .trustManager(tmf);
  .keyManager(new File("client.crt"), new File("client.key"));

SSLOptions sslOptions = new NettySSLOptions(builder.build());

Cluster cluster = Cluster.builder()
  .addContactPoint("127.0.0.1")
  .withSSL(sslOptions)
  .build();

Session session = cluster.connect(keyspace);
```

In contrast, the connection string for the Astra secured endpoint is as follows:

```
CqlSession session = CqlSession.builder()

.withCloudSecureConnectBundle(Paths.get("/path/to/secure-connect-database_name.zip
"))
  .withAuthCredentials("username","password")
  .build()
```

Note that we still provide a user and a password in the Astra connection. This allows us to take advantage of the fact that Astra provides both Cassandra role-based access controls (RBAC) and row-level access controls (RLAC). When an Astra DataBase is created a keyspace is specified during creation time. The set of Cassandra credentials provided by the account -- a limited superuser -- have full permissions for that keyspace and have the ability to create additional Cassandra roles with further-limited authorizations within that keyspace for operators, developers, or services.

Note: Astra as a platform uses native Kubernetes RBAC to govern pod and service account actions to enforce security at the platform level. This is not to be confused with Cassandra RBAC which is leveraged by DataBase users.

Administrators can also restrict specific access to rows (i.e. Cassandra partitions) using the RESTRICT keyword and GRANT access to Cassandra roles using a filtering string (see ***detailed docs on RLAC***).

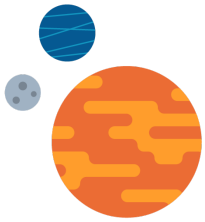
DataStax Access

The Astra platform is designed to securely house data across the widest range of global industries. The vision of the engineering and security teams is that Astra personnel will ensure data is highly available while distant from Astra personnel. By virtue of its cloud-native design, the majority of operations that are performed on customer clusters are performed by software rather than staff.

All-access to production environments is granted by role and monitored, with further restrictions governing access to compute instances running customer clusters.
No table-level access is required for troubleshooting, as recovery actions are via API.
Responsibilities are separated and logs are audited and used for automated alerts to detect unauthorized activities.

04

Astra Backup and Disaster Recovery



For business continuity and disaster recovery, Astra takes periodic backups of all customer clusters automatically and stores them in encrypted buckets in the respective cloud provider. In the summer of 2020, Astra will support multi-dc clusters for additional high availability in addition to the default behavior of spreading out Capacity Units across availability zones.