Couchbase

EBOOK

# Architecture Overview

Database-as-a-Service & Mobile App Services

# Contents

## INTRODUCTION

Capella is a modern distributed multi-model NoSQL Database-as-a-Service (DBaaS). Capella's core architecture supports a flexible JSON data model at its foundation and uses familiar relational and multi-model data access services to supply data to operational and analytic applications. Capella's advantages include fast in-memory performance, easy scalability, mobile app services, always-on availability, and advanced security.

These modern requirements have driven Couchbase's development from inception to:

- Ensure high-performance operations
- Provide data model and data access flexibility
- Support distributed cluster networks and mobility
- Provide incredible value and low TCO

### Database-as-a-Service

Capella is the fastest, easiest, and most affordable way to start with Couchbase. As a fully managed DBaaS, it automates setup, configuration, replication, and ongoing operations so you can focus on development and improve your time to market. And Capella puts industry-leading performance, flexibility, and cloud scalability at your fingertips.

### Mobile App Services

Capella App Services is a fully managed backend designed for mobile, IoT, and edge applications to guarantee they are always on, regardless of web connectivity and speed. Developers use App Services to access and sync data between Capella DBaaS and edge devices, and to authenticate and manage mobile and edge application users.

## DATABASE-AS-A-SERVICE

### Core database design

Couchbase developed its platform following three guiding principles: memory and network-centric architecture, workload isolation, and an asynchronous approach to everything.

**MEMORY AND NETWORK-CENTRIC ARCHITECTURE FOR SPEED AND LOW LATENCY**

- The most-used data and indexes are transparently cached in-memory for fast reads.

- Writes are performed in-memory and replicated or persisted synchronously or asynchronously. Transaction guarantees can be used to ensure consistency, but may introduce lags in performance.
- Internal Database Change Protocol (DCP) streams data mutations from memory to memory at network speeds to support replication, indexing, and mobile synchronization.

## MULTI-MODEL DATA ACCESS BLENDS JSON FLEXIBILITY WITH KEY-VALUE SPEED

Couchbase is a pioneer multi-model database that offers multiple data access methods to gain read and update access to its foundational JSON and key-value storage structures. Many other NoSQL systems have only one access method that is bound to their physical storage design structures on disk to minimize access latency.

Couchbase has multiple data access models including key-value, SQL++ query service, full-text search service, eventing service, operational analytics aggregation service, and backup service. In the Couchbase design, every access model can simultaneously utilize the cluster's data.

## WORKLOAD ISOLATION AND ASYNCHRONOUS PROCESSING

All databases perform different tasks in support of an application. These tasks include persisting, indexing, querying, aggregating, and searching data. Each of these workloads has slightly different performance and resource requirements. Couchbase's Multi-Dimensional Scaling (MDS) isolates these workloads from one another at both the process and node levels. MDS allows these workloads to be scaled independently from one another and their resources to be optimized as necessary. It allows the database to be performance-matched to the performance needs of the application, and the database to its available infrastructure.

For cloud deployments, it is advantageous from a cost perspective to red-line infrastructure instances before adding them, and to avoid idle and underutilized node instances. Couchbase transparently manages the topology, process management, statistics gathering, high availability, and data movement between these services.

Traditional databases increase latency and block application operations while running synchronous operations (for example, while persisting data to disk or maintaining indexes). Couchbase allows write operations to happen at memory and network speeds while asynchronously processing replication, persistence, and index management. Spikes in write operations don't block read or query operations, while background processes will persist data as fast as possible without slowing down the rest of the system. ACID transactions are available to the developer to ensure durability and consistency while data is in flight. Multiple transaction options allow the developer to decide when and where to increase latency in exchange for durability and consistency of transactions. Somewhat higher latency can be anticipated when multi-document and cross-collection transactions are implemented.

COUCHBASE IS A PIONEER MULTI-MODEL DATABASE THAT OFFERS MULTIPLE DATA ACCESS METHODS TO GAIN READ AND UPDATE ACCESS TO ITS FOUNDATIONAL JSON AND KEY-VALUE STORAGE STRUCTURES.

## JSON document data model

The JSON data model supports basic and complex data types, including numbers, strings, nested objects, and arrays. JSON provides rapid serialization and deserialization, is native to JavaScript, and is the most common REST API data format. Consequently, JSON is extremely convenient for web application programming.

A document often represents a single instance of an application object (or nested objects). It can also be considered analogous to a row in a relational table, with the document attributes acting similarly to a column. Couchbase provides greater flexibility than the rigid schemas of relational databases by allowing JSON documents with varied schemas and nested structures. Developers may express many-to-many relationships without requiring a reference or junction table. Subcomponents of documents can be accessed and updated directly, and multiple document schemas can be aggregated into a virtual table with a single query.

### JSON DOCUMENT FLEXIBILITY

In the Couchbase document model, a schema is the result of an applications structuring of its documents and their containment structures such as buckets, scopes, and collections. Schemas can be defined by application developers and managed by applications. This is in contrast to the relational model where the database (and the database administrator) manages the schema. Couchbase created the bucket-scope-collection-document organizational hierarchy (further explained below) to allow maximum flexibility in defining application data metamodels. A single JSON documents structure offers even more flexibility for the developer beyond the dynamic nature of scopes and collections. A JSON document's structure consists of its inner arrangement of attribute-value pairs. How the documents are designed or updated over time is up to the application developer. They can be normalized, denormalized, or a hybrid depending on the needs and evolution of the application. Using JSON, the developer can avoid the lengthy schema design, testing, and deployment cycles of traditional RDBMS-based systems.

A SINGLE JSON DOCUMENT'S STRUCTURE OFFERS EVEN MORE FLEXIBILITY FOR THE DEVELOPER BEYOND THE DYNAMIC NATURE OF SCOPES AND COLLECTIONS.

## Data access methods

Managing JSON data is at the core of Couchbase's document database capabilities, and there are several ways for applications to access the data.

| Access Method | Description |
|---|---|
| Key-value | An application provides a document ID (the key), and Couchbase returns the associated JSON or binary object. The inverse occurs with a write or update request. |
| Full text search | Using text analyzers with tokenization and language awareness, a search is done for a variety of field and boolean matching functions. Search returns document IDs, relevance scoring, and optional context data. |
| Query & analytics | SQL-based query syntax, similar to what is used with relational databases, interacts with JSON data and returns matching JSON results. Comprehensive DML, DQL, and DDL syntax supports nested data and nonuniform schema. |
| Eventing | Custom JavaScript functions are executed within the database based on timers or data changes. Accessing and updating data, writing out to a log, or calling out to an external system are all supported. |

COUCHBASE OFFERS A FLEXIBLE MULTI-LEVEL DATA CONTAINMENT AND ORGANIZATION STRUCTURE TO ORGANIZE DOCUMENTS, OPTIMIZE CLUSTER PERFORMANCE, AND FACILITATE HORIZONTAL SCALING.

## Organizing concepts for documents

Couchbase offers a flexible multi-level data containment and organization structure to organize documents, optimize cluster performance, and facilitate horizontal scaling. This data containment model consists of four levels: buckets, scopes, collections, and documents. This model maps easily to familiar RDBMS constructs of databases, schema, tables, and rows.

- **Buckets –** The topmost container in Couchbase is the bucket. One or many buckets can be defined and assigned to a Capella database.
- **Scopes –** Scopes are an intermediate data organization structure similar to a relational database schema. Scopes are defined by the collections of documents they contain or can access.
- **Collections –** Collections are categorical or logically organized groups of documents. The premise of collections is to behave as traditional table structures. Most group-oriented access activities are processed at the collection level to minimize full-database operations, simplify replication logic, and streamline indexing options.
- **Documents –** Documents are stored within buckets, but can also be organized within scopes and collections.

## Deployment design concepts

Services and nodes are key elements of the database architecture.

- **Services –** The core of Couchbase is the Data Service that feeds and supports all the other systems and data access methods. Multiple services that offer different types of data access or processing include Query, Indexing, Backup, Search, Analytics, and Eventing. A service is an isolated set of processes dedicated to particular tasks. For example, indexing, full-text search, and query are each managed as separate services. One or more services can be run on one or more nodes as needed.
- **Nodes –** Capella nodes are virtual machines that host single instances of Couchbase Server within a cloud service provider. Nodes can be added or removed easily through the Capella Control Plane and data is then automatically redistributed evenly across all nodes.
- **Database –** A database consists of one or more nodes running Couchbase Server. Nodes can be added or removed from a cluster. Replication of data occurs between nodes, and Cross Data Center Replication (XDCR) occurs between different clusters that are geographically distributed.

## Services

Each service has its own resource quotas, and where applicable, related indexing and inter-node communication capabilities. This provides several very flexible methods to scale services when needed. In addition to scaling up to larger machines or scaling out to more nodes, Couchbase also provides the ability to scale specific services independently from one another using multi-dimensional scaling. This MDS is the foundation for Couchbase workload isolation and is covered in more detail below.

Couchbase is different from other platforms where a monolithic set of services are installed on every node in a cluster. Instead, Couchbase uses a core data capability that feeds all the other services and a shared-nothing architecture that allows developer control over workload isolation. Small-scale environments can share the same workloads across one or more nodes, while higher scale and performance can be achieved with dedicated nodes to handle specific workloads. This provides the ultimate in scale-out flexibility. The cluster can be scaled in or out and its service topology can be changed on demand with zero interruption or change to the application.

Applications communicate directly with each service through a common SDK that is always aware of the topology of the cluster and how services are configured.

- **Data Service –** The Data Service, or key-value (KV) engine, is the foundation for storing data and must run on at least one node of every database. It is responsible for caching, persisting, and serving data to applications and other services. The cache provides consistent low latency for individual document read and write operations and streams documents to other services via Database Change Protocol (DCP). Due to their simplicity, KV operations execute with extremely low (often sub-millisecond) latency. The KV store is accessed using simple CRUD (create, read, update, delete) APIs, and provides the simplest interface when accessing documents using their IDs.

- **Query Service –** An engine for processing SQL++ queries. SQL++ combines the flexibility of JSON with the expressive power of SQL. It provides a rich set of features and familiar data definition language (DDL), data manipulation language (DML), and query language statements, but can operate in the face of NoSQL database features such as key-value storage, multi-valued attributes, and nested objects. Also, users can define ACID transactions within SQL++ for one or more documents across collections and nodes. Transactions in SQL++ have adopted a nearly identical syntax to SQL for relational databases. The Query Service uses a cost-based query optimizer, patented in 2021, to take advantage of indexes that are available.
- **Index Service –** Indexing is an important part of making queries run efficiently and self-update as data mutates. This service supports multi-index types and includes an Index Advisor that recommends specific indexes to build based upon query statements and data structure.
- **Search Service –** An engine for performing full-text searches on stored JSON data. Users can create and query inverted indexes for searching of free-form text within a document. Customers using the search service often no longer need a third-party search tool.
- **Eventing Service –** Eventing supports custom server-side functions (written in JavaScript) that are automatically triggered using an event-condition-action model. These functions receive data from the DCP stream and execute code when triggered by data mutations. This service offers a feature like the change data capture found in event handlers, and also offers a feature similar to the multi-channel data streaming found in solutions such as Apache Kafka.
- **Analytics Service –** This service provides an ad hoc querying capability without the need for indexes. It uses the same SQL++ language as the query service, and uses a hybrid operational and analytical processing model for real-time operational analytics on active JSON data within Couchbase. The service efficiently runs complex queries over a large number of documents and includes features such as ad hoc joins, set, aggregation, and grouping operations with efficient parallel query processing and bulk data handling.

## Distributed design

Capella's distributed nature makes high availability, scaling, and disaster recovery easier.

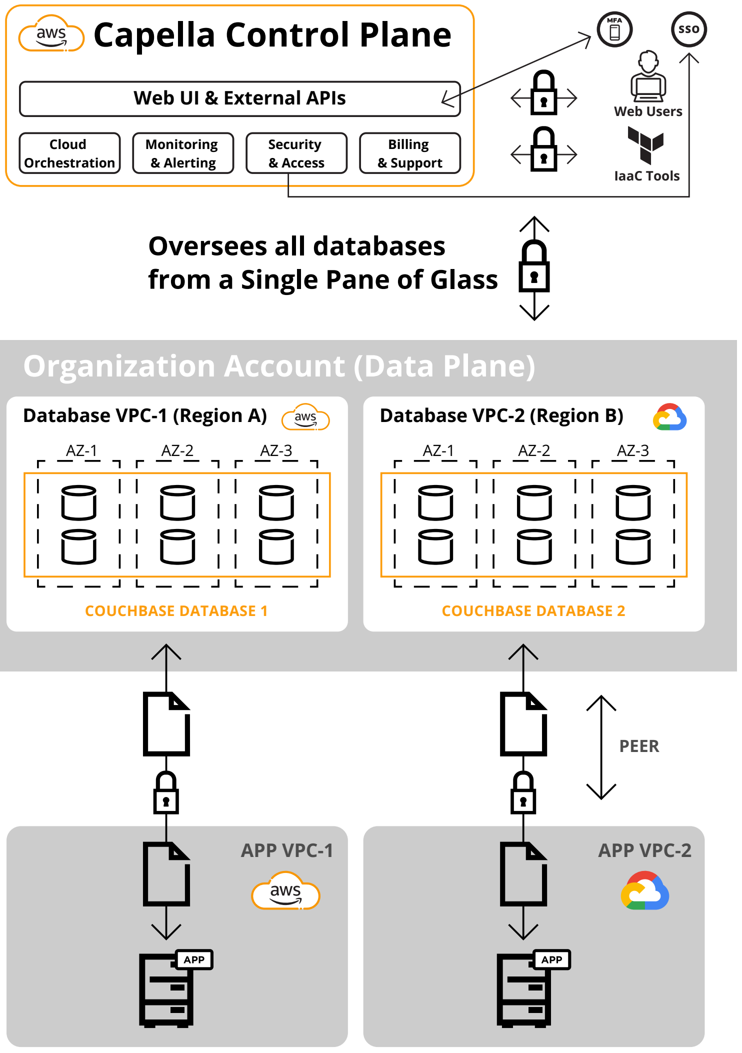
- **Data distribution –** Couchbase automatically partitions and replicates data into vBuckets (synonymous to shards) to automatically distribute data across nodes. This helps enable data replication, failover, and dynamic database reconfiguration. Because vBuckets do not have a fixed physical location on nodes, they are mapped to nodes in a cluster map. Through the Couchbase SDK, the application automatically accesses data without needing to know the exact location of the data.
- **Data transport via DCP –** As data mutates, in-memory replication is used to maintain data updates within Capella and to external services such as Spark or Kafka that are fed from the DCP stream.

- **Multi-dimensional scaling –** You can improve performance and throughput for systems by independently scaling services to match workloads. Scale-out and scale-up are the two scalability models typical for databases, and Couchbase takes advantage of both. You can combine and mix these models in a single database to maximize throughput and minimize latencies.
- **Failover –** If a node fails in Capella, replicated data on other nodes are promoted to active. A new node is then automatically provisioned and data is rebalanced across all the nodes.

## AS-A-SERVICE-ASPECTS

### Architecture

Capella's core architectural aspect is the split of the web UI Control Plane designed for management and the Data Plane for data storage.



### CONTROL PLANE

The Control Plane is a web UI that manages the cloud orchestration Infrastructure-as-a-Service (IaaS), monitoring, alerting, security, access, billing, and support capabilities. It's the access point for your organization's users and also allows access to infrastructure-as-code (IaC) tools such as Terraform. Backend services can be accessed by REST-based tools via a management API.

### DATA PLANE

The Data Plane is where you manage your Capella databases. A database resides in a single region (distributed across multiple availability zones) within a single cloud service provider (CSP), but the Control Plane can control multiple databases across various cloud service providers. Furthermore, data can be replicated between databases, with that replication configured within the Control Plane. From a security perspective, the Data Plane has no internet access unless IPs are specifically allowed or a connection is established through VPC peering or an AWS PrivateLink connection. Disk data, backups, and all traffic is encrypted.

## Management

### USERS, PROJECTS, AND RBAC

An organization is the top-level organizational element for managing users, projects, and databases. By default, organizational members do not have data access, but instead have access to the Control Plane. A project is used to organize groups of databases. People must be added to projects and assigned access to databases within a project. Organizations can also add SSO groups (teams) with a project role to access databases within a project.

### SINGLE SIGN-ON (SSO)

SSO is a convenient way to maintain users by making use of their existing corporate credentials. Authentication is delegated to the SSO provider (Azure AD and Okta are supported). Provisioning new users via SSO eliminates the overhead of having to send invites.

## Deployment

### DATABASE DEPLOYMENT

When deploying a database, you must choose a CSP, a region, and a CIDR block. (A default is provided, but can be changed before deploying.) Couchbase is constantly adding regions, and up-to-date regions can be found on the Couchbase Documentation pages for AWS, GCP, and Azure. You can select the version of the Couchbase Server you would like to use and assign services to nodes. These services can be changed after deployment. You can also choose your support plan and availability mode (single or multi-availability zones), and can choose to purchase credits on a prepaid or pay-as-you-go basis.

### STORAGE ENGINE

Capella supports two different backend storage mechanisms, which are set per bucket. A single Capella database can have a mix of Couchstore and Magma buckets.

- **Couchstore –** Couchstore is the default bucket storage engine that has been in use for more than 10 years. It's optimized for high performance with large datasets while using fewer system resources. (The minimum bucket size for the Couchstore backend is 100 MiB.) If you have a small dataset that can fit in-memory, then you should consider using Couchstore.
- **Magma –** Capella's latest storage engine is designed for high performance with very large datasets that don't fit in-memory. It's ideal for use cases that rely primarily on disk access. The performance of disk access will be as good as the underlying disk subsystems. Magma can work with very low amounts of memory for large datasets (e.g., for a node holding 5 TiB of data, Magma can be used with only 64 GiB RAM). It's especially suited for datasets that won't fit into available memory.

You can learn more about Capella's storage engines in our Couchbase documentation.

## Development

Couchbase provides several tools for developers:

### PLAYGROUND

This tool is integrated into Capella and comes with an SDK tutorial and a SQL++ tutorial. The SDK playground offers examples of multiple SDK languages. SQL++ gives examples of the Couchbase query language. Both tutorials are designed to guide a new developer through several chapters from basics to more advanced concepts.

### QUERY WORKBENCH

The Query Workbench allows users to access data via SQL++ and see data in JSON and tabular formats. The tool provides a built-in index advice feature that tells users what indexes are needed to optimize queries. Inverted search indexes can be created to support search, and JavaScript-based user-defined functions can be used to manipulate data.

### COUCHBASE SHELL

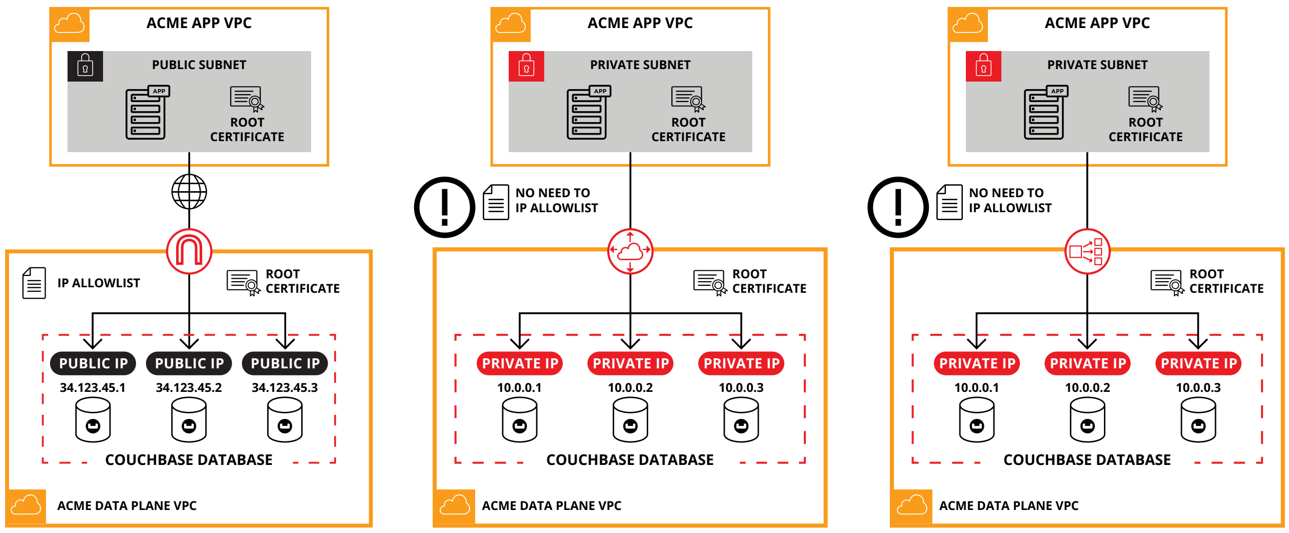
Couchbase Shell (cbsh) is a modern, productive shell that provides CLI access to Capella. It can be obtained via GitHub.

## Connecting

**OPTION 1: PUBLIC CONNECTION**
- Connect to Node **Public** IP
- Needs IP Allowlisting
- Traffic traverses public Internet

**OPTION 2: VPC PEERING**
- Connect to Node **Private** IP
- Traffic contained within CSP backbone
- Joined network between the 2 VPCs

**OPTION 3: PRIVATE LINK**
- Connect to Node **Private** IP
- Traffic contained within CSP backbone
- Unidirectional and very secured



### COUCHBASE SDKS

Capella works with the latest versions of all supported Couchbase SDKs. Developers can choose from over 10 SDKs of their favorite programming languages.

### CONNECTORS

To exchange data with other platforms, we offer various big data connectors for products like Kafka, Spark, and Elasticsearch.

### REST API

Couchbase provides a series of RESTful APIs that enable you to integrate with Capella to perform operations such as:

- Onboarding and offboarding users
- Managing the lifecycle of a cluster
- Getting monitoring information for a cluster

### APPLICATION CONNECTION

For connecting applications to Capella, you have several options:

- **Public connection –** This is the simplest option and requires the use of IP addresses for encrypted data to traverse the public internet. Public connections should not be used for production environments.
- **VPC peering –** The CSP backbone contains both the connection and traffic. This option reduces networking costs compared to a public connection.
- **AWS private endpoint –** Allows Capella to be offered as a private service and functions as if it were hosted directly within a team's Amazon VPC. This allows access to a specific service or application, and only private endpoints can initiate a connection.

CAPELLA PROVIDES A WIDE VARIETY OF METRICS TO MONITOR AND HELP OPTIMIZE PERFORMANCE.

## Operations

### SCALING

Capella makes it easy to evolve your configuration. You can add or remove nodes at any time and change the amount of RAM, vCPUs, disk space, and type of disk volume (general purpose or high performance). Changes are made with a few clicks, and Capella automatically rebalances data to the new configuration.

### MULTI-DIMENSIONAL SCALING

MDS allows you to further optimize your configuration as application needs evolve over time. MDS allows workloads to be scaled independently and hardware usage to be optimized to help drive down total cost of ownership.

### GEO-REPLICATION

Capella's Cross Data Center Replication (XDCR) technology replicates data between databases in different regions. XDCR provides an easy way to replicate active data in-memory to multiple geographically diverse data centers either for disaster recovery or for high availability. It can be set up on a per-bucket or per-collection basis and can be unidirectional or bidirectional. It also provides built-in conflict resolution if the same document was mutated on a separate database before it was replicated.

### BACKUP AND RESTORE

A robust scheduled backup and retention policy is recommended as part of an overall disaster recovery plan for production data. Backup is done on a per-bucket level and can be scheduled on monthly, weekly, or daily cycles, or on demand. You can set up both full and incremental backups. Restoring data can happen at the bucket or collection level with filtering options. Additionally, data can be restored into Capella from a self-managed Couchbase Server environment.

### MONITORING AND ALERTING

Capella provides a wide variety of metrics to monitor and help optimize performance. Metrics can be viewed within a Capella Control Plane configurable dashboard or incorporated into your Prometheus tool. Capella has a set of conditions that generate alerts and provide actionable suggestions when a threshold is hit (e.g., a suggestion to increase RAM or disk size).

### MAINTENANCE AND UPGRADES

If you want to upgrade your databases, those processes can be scheduled within the Control Plane. Security updates are forced immediately. Notifications about upgrades are sent via email and within the UI.

## Security

Couchbase supports the most critical and sensitive workloads for industry-leading businesses every day. Capella's security architecture is based on industry best practices for security and three key pillars: Verify explicitly, least privilege, and platform monitoring. You can learn more and get detailed security whitepapers and information about compliance at our Trust Center.

The foundation of security in a cloud database is a hardened environment that removes nonessential software, roles, and ports while leveraging an IaaS provider's alerting and auditing services. Trusted and immutable operating system (OS) images are used to protect the OS, with verification upon deployment and ongoing scanning for vulnerabilities afterward. Additionally, end-to-end configuration is automated via templates to ensure consistency. Monitoring is also in place to identify potential misconfigurations.

## NETWORKING SECURITY

By default, the Data Plane only allows clusters to connect to trusted IP addresses that have been defined within the Control Plane. Any attempted connection from an IP address not in a cluster's list of allowed IP addresses will be denied. With VPC peering, traffic never crosses the public internet, which reduces threat vectors and DDoS attacks. If you're using AWS for your Data Plane, you can further enhance security by using PrivateLink to contain traffic within the CSP backbone with unidirectional access. Alternatively, you can set up Capella as a private service that functions as if it were hosted directly within a team's Amazon VPC. This allows access to a specific service or application, and only private endpoints can initiate a connection.

## ACCESS SECURITY

To bolster security access, Capella is designed so that the Control Plane and Data Plane live in separate VPCs. Access to data is separate from access to the Control Plane, and specific credentials must be established for application access. All admins, users, and applications must authenticate in order to gain access and then be authorized with specific access rights. Multi-factor authentication is possible and recommended.

## DATA SECURITY

Data is encrypted in transit and at rest. In transit, data is encrypted via TLS, which cannot be turned off. If you want to extend data storage encryption within the database, this can be done at the field level within JSON documents. Also, backup data is written to encrypted disks using the cloud provider's native encryption process. Capella creates, manages, and controls cryptographic keys using a CSP's key management system (e.g., KMS for AWS).

## VULNERABILITY MANAGEMENT

Capella protects against threats like brute force attacks, rate-limiting attacks, cross-site request forgery, and more. Capella maintains centralized logs securely and alerts Couchbase site reliability engineers (SREs) of operational concerns should they arise. To reduce potential vulnerabilities, patching is automated and includes monitoring alerts and management reviews. Couchbase has established a formal Incident Response Policy to inform you in the event of a security-related event.

## APP SERVICES FOR MOBILE AND IOT

### About Capella App Services

App Services provides a hosted gateway for bidirectional data synchronization between Capella and embedded apps on smartphones, tablets, IoT devices, and custom embedded devices. It works in tandem with Couchbase Lite, the embeddable version of the Couchbase database. Wherever Couchbase Lite runs, App Services can securely sync the data it captures to Capella buckets and other embedded devices.

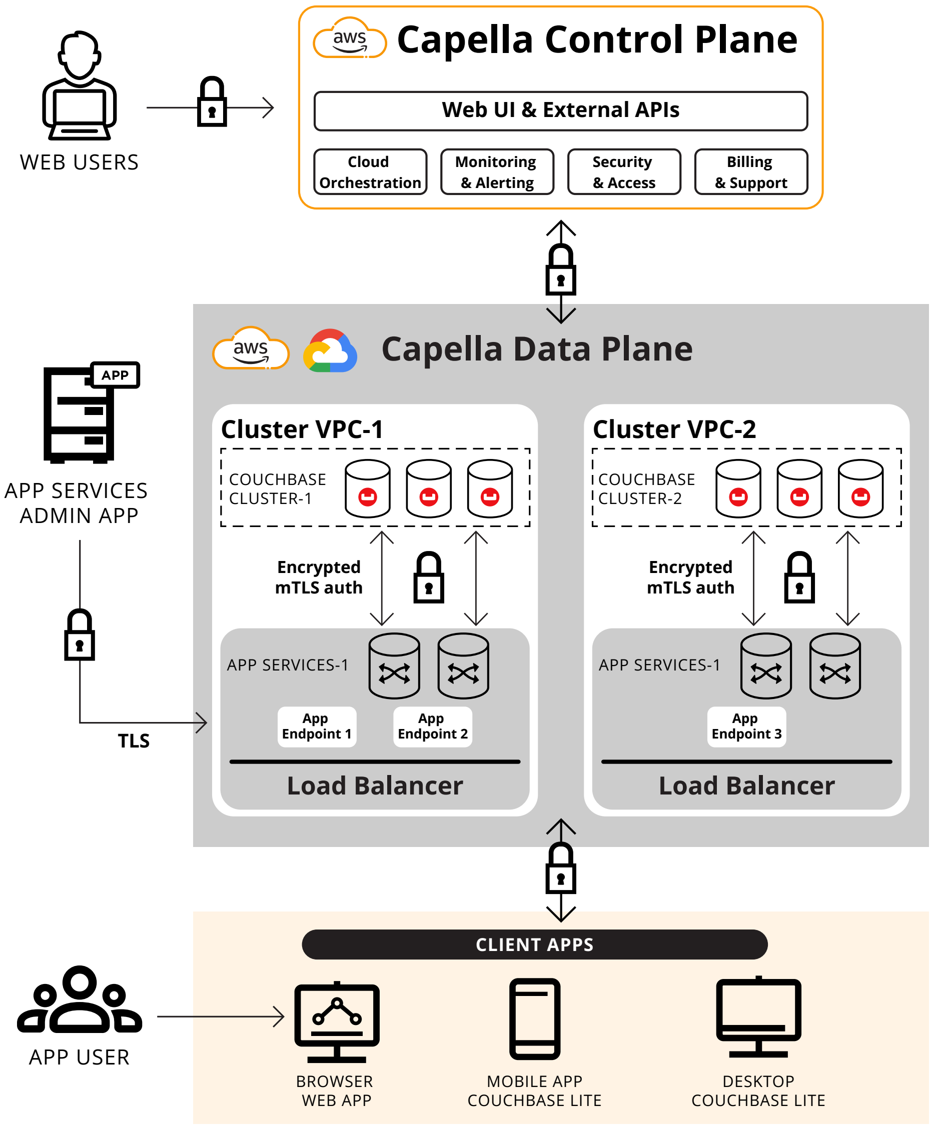
App Services also manages secure data access with role-based access control, providing authentication for mobile users. These key capabilities in Capella are offered as a ready-to-use service for mobile and IoT developers, making it faster and easier than ever to build highly performant and reliable applications.

### About “Offline-First” applications

For applications that need to operate in areas with slow or no internet, embedded Couchbase Lite provides on-device data storage and processing. This allows apps to work all of the time, whether on or offline (hence the term offline-first). App Services sync is smart enough to know when connectivity is interrupted. When it’s restored, App Services can automatically start syncing from where it left off even after long periods of time.

More importantly, when multiple Couchbase Lite clients are close to one another, but have no internet, they can still do peer-to-peer syncing. This is a feature unique to Couchbase that enables offline-first collaboration without the need for any central control point.

## App Services architecture



When you create an App Service and associate it with a Couchbase Server cluster, you are effectively extending or enabling it for data sync. A Couchbase Server cluster can only be linked to one App Service.

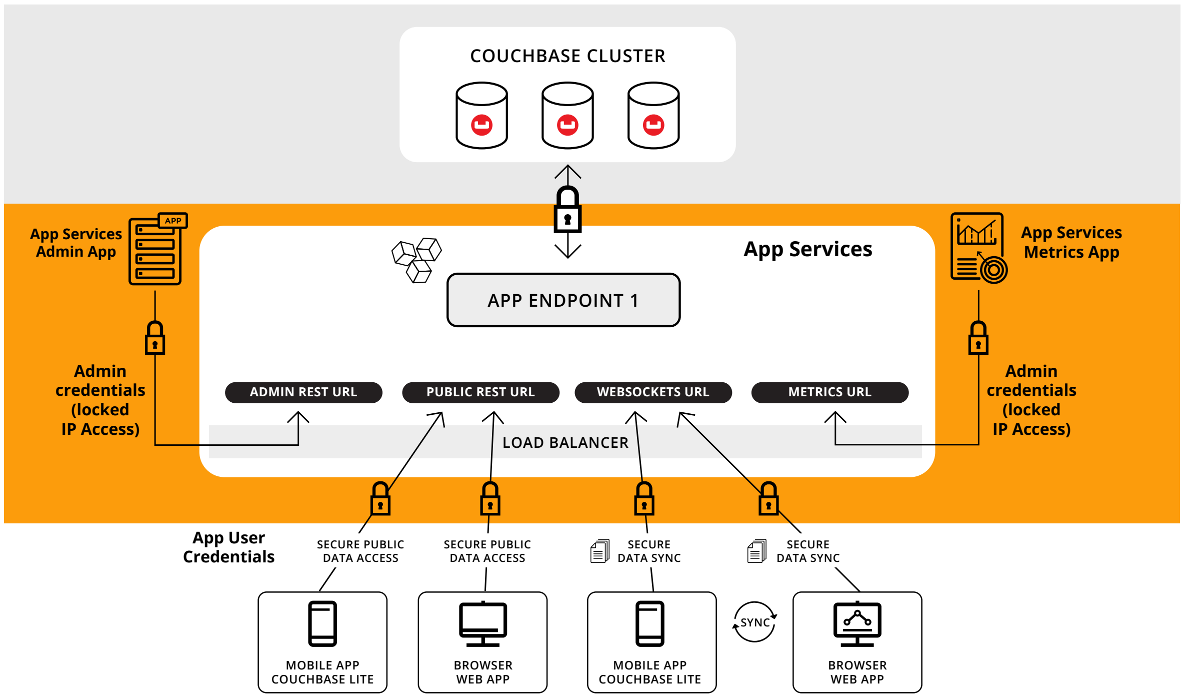
When an App Service is created, a cluster of Sync Gateway nodes is deployed behind the scenes in the same VPC network as the corresponding server cluster. Communication between the App Services cluster and the central Couchbase Server cluster is secured using TLS and x.509 cert-based authentication. The Sync Gateway cluster is fronted by a load balancer that balances incoming client requests across the App Services nodes.

An App Service can handle multiple client applications, each represented by an App Endpoint. Conceptually, an App Endpoint represents the instance of your application on the App Service. Each App Endpoint is backed by a server bucket. If you have multiple applications, each will have its own App Endpoint.

Mobile, desktop, and web client apps can access and sync data by connecting to the corresponding App Endpoint.

## App Endpoint connection points

There are multiple options for connecting clients to an App Endpoint. Your choice depends on the type of application and use case.



### SECURE WEBSOCKETS PUBLIC URL

Offline-first sync is the ability for apps to run in offline mode in the face of temporary or extended network disruptions and to sync data with the backend servers when connectivity is restored. Mobile, desktop, and embedded apps powered by Couchbase Lite can locally store and access data in disconnected mode and sync data with App Services when there is connectivity. With the internet being inherently unreliable, the use cases for offline-first data sync are vast and varied.

### SECURE PUBLIC REST API

Applications can also access data securely over a public REST Endpoint. This is useful when there is reliable network connectivity and no need for offline storage, or when the apps are running on hardware that doesn't have local storage for running a local embedded database like Couchbase Lite.

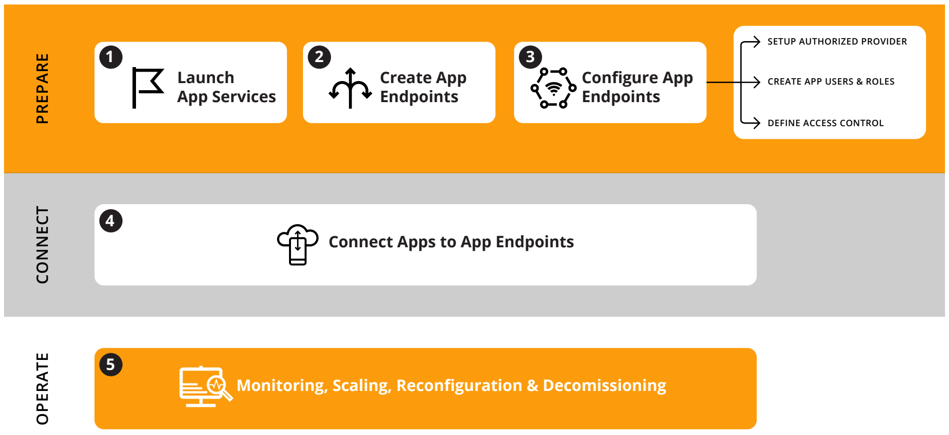
### SECURE ADMIN REST API

Administration applications can be granted authenticated access to the Admin REST API in order to programmatically create and manage users, roles, and sessions. Admin Apps are typically hosted in the cloud backend. An example of an Admin App is a login service that handles custom authentication and is responsible for registering users via the secure admin REST API following successful user authentication.

### SECURE METRICS REST API

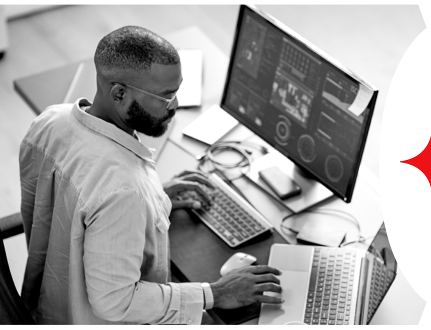
Monitoring frameworks like Prometheus can access stats exposed via the metrics REST Endpoint. In addition, App Services also supports a dashboard of common operational stats.

## User journey


PREPARE
1 Launch App Services
2 Create App Endpoints
3 Configure App Endpoints
SETUP AUTHORIZED PROVIDER
CREATE APP USERS & ROLES
DEFINE ACCESS CONTROL
CONNECT
4 Connect Apps to App Endpoints
OPERATE
5 Monitoring, Scaling, Reconfiguration & Decomissioning

Prerequisite: App Services requires a Couchbase Capella server cluster. Follow these steps to create a Capella server cluster and set up a bucket.

## Prepare



### LAUNCH APP SERVICES

When you create an App Service and associate it with a server cluster, you are effectively enabling it for data sync. When creating an App Service, you give it a name, designate an associated Capella cluster, then choose the deployment configuration that includes the correct number of nodes and type of computer (RAM/core).

### CREATE APP ENDPOINTS

App Endpoints represent the instance of your application on an App Service. You can create multiple App Endpoints on an App Service, each backed by a unique bucket in the corresponding Couchbase Server cluster. By default, all documents in the corresponding bucket are imported by the App Endpoint.

### CONFIGURE APP ENDPOINTS

When the App Endpoint is created, it is set up in offline mode. This allows users to complete the security configuration of the App Endpoint before exposing it to applications.

### AUTHENTICATION PROVIDER

Authentication providers define how users are authenticated with the App Services. A default auth provider of basic auth is selected for you during App Endpoint creation. So you can skip this config if the default option works for your application.

Capella supports the following modes of authentication:

- ***Basic Auth* –** This is where the app users are authenticated using username and password credentials that are Base64 encoded and passed in as part of the authorization header of an HTTP request.
- ***Open ID Connect (OIDC)* –** App users are authenticated against a third-party identity provider that is registered with App Endpoint. This is implemented using OIDC Implicit flow.
- ***Anonymous* –** In this mode, we allow unauthenticated read-only access to data. This mode can be useful when your app is only dealing with public static data.

### USER MANAGEMENT

With the exception of "Anonymous" mode, all client-side access must be authenticated with suitable user credentials. The choice of how users (and roles) are created depends on the Authentication Provider that is configured.

- ***Basic Auth* –** Users are created via the Capella web UI or via Admin REST Endpoint.
- ***Open ID Connect (OIDC)* –** By enabling the "auto-register" option when configuring OIDC provider, users will be automatically created on App Service after successful authentication.

### ACCESS CONTROL

Access control is implemented using the channel-based access control model of Couchbase Mobile. Access control specifies who has access to what data. This is specified via a JavaScript access control function. Read access control is at the granularity of a document, while write access control is at the granularity of a field.

## Connect

After completing the security setup for the App Endpoint, unpause the App Endpoint to bring it online. Once online, apps can be connected using any connection points discussed earlier.

## Operate

Once your App Service is operational, you can administer the App Service and App Endpoints and change the configuration to meet the evolving needs of the apps.

### MONITORING

Metrics dashboards provide insights into resource utilization of the App Service as well as the operational state of the App Endpoints. These include stats such as the number of documents read/written, error counts, number of active replications, etc.

### ACTIVITY LOG

All key system events of type info, warning, and error are recorded in the activity center. Users are also alerted to key events that may need attention, such as significantly high memory utilization over an extended period of time.

### ON-DEMAND SCALING

To keep up with the evolving needs of the app, users can scale App Services horizontally and/or vertically by changing the number of nodes and/or compute type.

## SUMMARY

Capella is a modern distributed multi-model NoSQL DBaaS. It includes fast in-memory performance, easy scalability, mobile App Services, always-on availability, and advanced security. At the heart of Capella is a flexible JSON data model, and it uses familiar relational and multimodel data access services to supply data to modern applications. Capella's architectural benefits include:

- High-performance operations
- Data model and data access flexibility
- Powerful solutions for mobile and IoT applications
- An incredible value that lowers TCO

### Resources

- Couchbase Capella product page
- App Services page
- Capella free trial
- Capella Documentation
- Capella Trust Center

Couchbase

Modern customer experiences need a flexible database platform that can power applications spanning from cloud to edge and everything in between. Couchbase's mission is to simplify how developers and architects develop, deploy and consume modern applications wherever they are. We have reimagined the database with our fast, flexible and affordable cloud database platform Capella, allowing organizations to quickly build applications that deliver premium experiences to their customers—all with best-in-class price performance. More than 30% of the Fortune 100 trust Couchbase to power their modern applications.

For more information, visit www.couchbase.com and follow us on Twitter.