



# Security Checklist

## for Self-Managed Weaviate Deployments

This document provides a list of security measures that you should implement to protect the confidentiality, availability and integrity of your Weaviate installation. The list is not meant to be exhaustive.



## Production Checklist:

### ✓ Enable OIDC Authentication

- ✓ Weaviate supports authentication via any OIDC provider (such as Okta, Microsoft or Apple), or your own OIDC token server, such as Dex or Keycloak. Both the Weaviate server endpoint and our clients can support the authentication flow.
- ✓ OIDC allows you to authenticate external users with Weaviate, without needing to issue them with a separate account or API key. Weaviate will not have access to those users' passwords.
- ✓ Our [authentication documentation](#) details the configuration steps required to enable and successfully test OIDC.

### ✓ Disable anonymous access

- ✓ By default, Weaviate allows anonymous users to query the database. Disable this feature by configuring `AUTHENTICATION_ANONYMOUS_ACCESS_ENABLED: 'false'`. With this feature disabled, anonymous access will return 401 unauthorized.
- ✓ Review our [documentation here](#) for more details.

### ✓ Configure Role-based Access Control

- ✓ Weaviate supports a number of roles for users to adopt when they log in. Both Admin (read and write), and Read-only roles are supported and can be defined per user within the Weaviate configuration file.
- ✓ Review our online [documentation here](#) for information on how to enable and configure RBAC.
- ✓ Support for more Granular RBAC permissions are expected to launch by Q2 2025, please watch our releases page for more information.
- ✓ Review our [documentation here](#) for more details.


### ✓ Limit Network Access

- ✓ By default when deployed via Helm, Docker or built from source, Weaviate uses port 8080 for communications over http.
- ✓ In addition, Weaviate's Helm charts and Docker images include a local Grafana instance that runs on port 3000 with a default login.
- ✓ We recommend that the server(s) that Weaviate will run on are configured with a firewall and restrictive rules to limit access to known IP addresses.
- ✓ In addition, we recommend that the Grafana default credentials are changed.

## ✓ Encrypt Data at Rest

- ✓ In order to safeguard data in Weaviate, we recommend that the data is stored on buckets/disks with encryption enabled.
- ✓ When using storage buckets from Amazon, Google and Microsoft, check that KMS encryption is enabled.

## ✓ Configure Data Backups to a separate location

- ✓  Remember that system backups triggered by the Weaviate client do not include inactive or offloaded tenants.
- ✓ We recommend that Weaviate's built-in data backup tool is used to create copies of your data at regular intervals. [Our Backup Module](#) includes the ability to natively integrate with Amazon, Google and Microsoft's storage solutions.
- ✓ Alternatively, for more complex deployments, Weaviate can back up to a local filesystem, which can be any underlying storage platform provided it is presented to Weaviate as an accessible path (for example via a persistent volume or bind mount).
- ✓ Backups initiated via the Weaviate clients or the API will report a status (successful or unsuccessful) which can be used with an external monitoring tool to trigger an alert if a backup were to be unsuccessful.

## ✓ Monitor System Activity

- ✓ Although Grafana is included in the Weaviate Helm chart and Docker images, we recommend that in production Weaviate is integrated with an existing monitoring or observability solution, such as Grafana Cloud, Datadog, Nagios or PRTG.
- ✓ Configuration of external monitoring systems is outside of the scope of this checklist, however we provide some [deployment guidance here](#) which will help integrate with any external tool.

## ✓ Encrypt Communications (with TLS)

- ✓ When deploying Weaviate in a production environment, it is important to encrypt network traffic using TLS or similar.
- ✓ We recommend that either a load balancer, proxy or service mesh gateway is deployed in front of the Weaviate service with a valid certificate, and that only TLS1.2 is enabled with support for AES-256 ciphers.
- ✓ If encryption inside the Weaviate cluster or infrastructure is required, then we recommend a service mesh supporting mTLS or similar is deployed.
- ✓ Remember to set the HTTP Origin in the Weaviate configuration file when this change is made. Review our [documentation here](#) for guidance.

## ✓ Enable Replication for high availability

- ✓ Weaviate supports replication to improve performance and as part of a high-availability strategy. Replication is available on a collection or across the entire Weaviate deployment.
- ✓ When deploying in a cloud environment, we recommend that replication is configured across at least two availability zones, and if the use case is mission-critical, across multiple regions.
- ✓ Note that replication is disabled by default and must be configured for your use case.
- ✓ Our [Replication Documentation](#) illustrates the process of configuring replication and how to set consistency levels for data.

## EDR/Anti-Malware Scanning:

- ✓ Please note that as our application is written in go-lang, some commercial antivirus products may falsely alert when scanning the binaries.
- ✓ For performance reasons, we recommend excluding the Weaviate data store from on-access scanning.

## Security Contacts:

Please contact us via [hello@weaviate.io](mailto:hello@weaviate.io) if you discover a security issue or vulnerability affecting our application, platform or services.

## Ongoing Checks:

### ✓ Update Weaviate

- We recommend that customers always use a supported version of Weaviate, which is shown on our Github repository. We regularly patch our software to add new features or when a vulnerability is discovered.
- For more information about the security and feature content of our releases, review our release notes on Github.

### ✓ Patch Servers

- In addition to the Weaviate software, it is important to ensure that servers and storage systems that host Weaviate are patched to the latest versions supported by the vendor.
- If Weaviate is being run in a Docker or Kubernetes environment, then ensure that any service mesh, security, proxy or load balancer appliances are also kept up to date.

### ✓ Review Rules

- As part of ongoing security, we recommend that rules governing access to Weaviate are regularly reviewed and updated. This includes firewalls with CIDR or IP based restrictions, users with privileged access to servers, storage or the application and any access to system and data backups.

### ✓ Rotate API Keys

- Finally, the API keys generated by Weaviate should be considered ephemeral and rotated regularly in line with industry best practices.
- We do not recommend that API keys are hard-coded into clients or integrated software, as the theft or loss of those keys could grant an attacker privileged access to all Weaviate data.