Use Redis with Identity Server

Redis is supported by Ubisecure SSO as of version 8.3

Introduction

From redis.io

> *Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker. It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs, geospatial indexes with radius queries and streams. Redis has built-in replication, Lua scripting, LRU eviction, transactions and different levels of on-disk persistence, and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster.*

This article and related sub-articles cover the topic of using Redis as a session storage for Ubisecure Identity Server, namely Ubisecure SSO as Ubisecure CustomerID does not use Redis at the moment. This document is relevant when planning a high-performance deployment of Ubisecure Identity Server.

For configuration and installation, please refer to

- How to install and configure Redis for Identity Server

Deployment models

Ubisecure Identity Server can be deployed using three different models

- Single node
- High Availability
- High Performance

This article relates to the High Performance deployment where Redis is included.

References and links

Anchor	Link	Description
[1]	https://redis.io/	Redis homepage
[2]	https://redis.io/topics/quickstart	Redis quick start and installation
[3]	https://redis.io/topics/cluster-tutorial	Redis clustering guide
[4]	https://redis.io/topics/admin	Topics around operating Redis
[5]	https://redis.io/documentation	Redis documentation

Terminology

Term	Description
Active node	A process like a web server serving incoming requests.

Execution environment	A physical or virtual runtime environment where processes run. For example, physical servers, virtual machines, container orchestration etc.
Primary / Replica	The primary instance is the main source whereas replica is mirroring the primary source and waiting to take over in case of failure.
Passive node	A process like a web server that is ready to server incoming requests but is not actively doing it i.e. is in standby mode.

Addressing linear scalability with Redis

As mentioned, Ubisecure Identity Server can be deployed to high-availability or high-performance environments. When the anticipated maximum load of incoming authentication requests (logins) will exceed the advised capacity of a High-Availability environment then a High-Performance deployment model should be considered.

Currently the rough estimation of transactions per deployment type is as follows

Deployment model	Active SSO nodes	Passive SSO nodes	Number of login transactions / sec	Average response time
High-Availability (active-passive)	1	1	30	2.5 s
High-Performance (2 active nodes)	2	0	80	1.5 s

The metrics are based on Ubisecure's reference environment. The test case is for performing end-user login initiated by OpenID Connect Authorization code flow and requesting UserInfo.

High Availability deployment is meant for systems where failure of a component does not cause a service break. High Performance deployment model guarantees higher rate of login transactions as frequent write-intensive operations are performed on Redis instead of Ubilogin Directory. In this deployment model, Ubisecure SSO can serve requests using multiple instances, each being in active state.

Failure tolerance

Replica election process

In a multi-node Redis deployment, failure tolerance is achieved through multi-master / multi-replica deployment where replicas take over primary nodes when those go down.

The replica election process is roughly as follows

- A replica election starts when a primary node is in failed state.
- A replica whose primary node is down starts the election.
    - In case of multi-replica setup, any of the replicas can start the election.
- Primary nodes participate to the election and vote for a replica to be promoted as primary.
- When a specific replica receives votes from majority of primary nodes it wins the election.

Number of primary nodes in a cluster

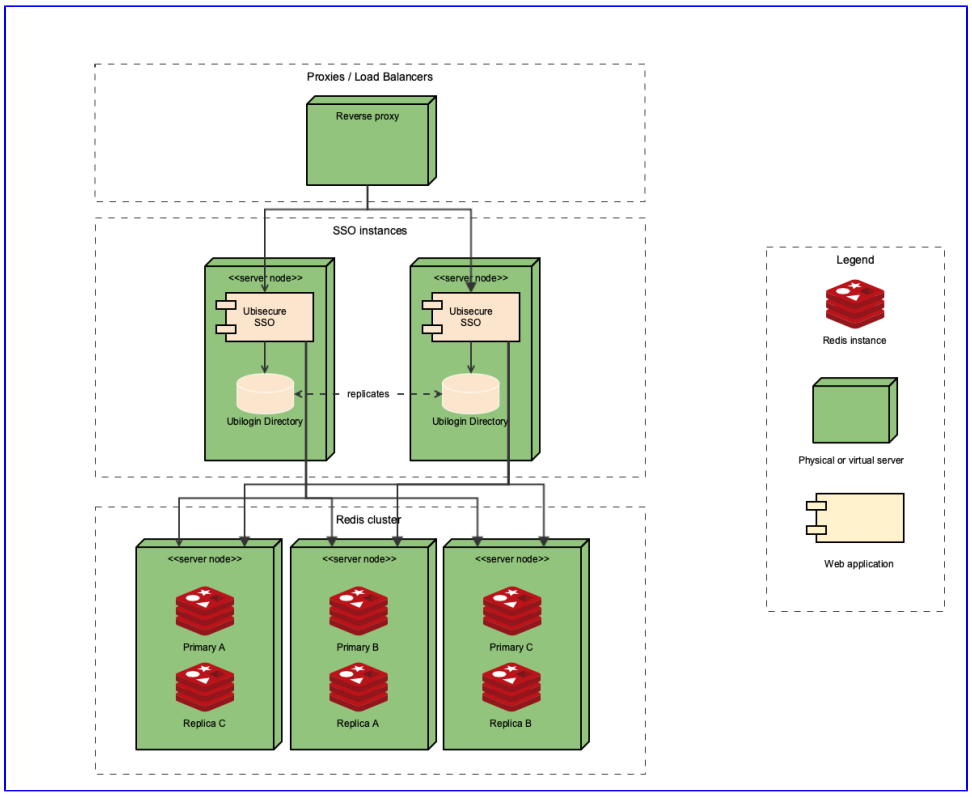
Note that it is essential that the majority of primary nodes are alive in order to the cluster to be considered healthy and capable of performing replica election. Therefore the suggested number of primary nodes is an odd number.

Voting fails if the majority of primary nodes are down or the vote is equal

More detailed replica election process is described in the Redis Cluster specification.

Reference Architecture

The following diagram depicts Ubisecure's Reference Architecture for high performance deployments with Redis.



In this deployment model, Ubisecure SSO instances operate in **active** mode, serving incoming requests from external clients, through a proxy or a set of proxies. It is suggested that Ubilogin Directory instances reside close to SSO instances to minimize latency as LDAP still acts as a configuration storage.

The deployment units should reside close to each other to minimize latency and issues in the case of network partitioning.

Proxies

Depending on the deployment model, the system is accessed through proxies or load balancers. Depending on the High-Availability requirements there can be many instances. This is out of the scope of this document but the general assumption is that the proxy layer can delegate traffic to all SSO instances that are active in the system.

> **HTTP sticky sessions**
>
> During for example login transaction, the browser communicates to Ubisecure SSO via an active HTTP session, therefore the proxy must make sure that HTTP sessions are sticky and requests in an HTTP session are delegated to the same SSO instance.

SSO instances

Each of the SSO instances are able to process incoming requests in this deployment model. The reference architecture consists of two active instances but for linear scalability, the number of instances is not limited. However, the general suggestion is to start small and increase when there is a need to process more incoming requests.

In this deployment model, LDAP acts mostly as a configuration storage. It must also be noted that LDAP instances must replicate the data as the data that is read from LDAP contains:

- Configurations for application integrations
- Configurations for authentication methods
- Authorization policies
- SSO Management and API users and groups
- Access and Refresh Tokens
- Persistent IDs

Users in Ubilogin Directory

When using Ubilogin Directory as a user repository, the user last login timestamp is still updated i.e. some write operations are still performed in LDAP.

Redis cluster

Redis acts a Single-Sign-On session storage.

In order for a Redis cluster to be operational and failure tolerant, at least six Redis instances are needed. This means three primary instances and one replica for each. When increasing the number of primary instances, it is suggested to use an uneven number in order for the replica election process to be effective. As mentioned above, the election process only works if the majority of primary instances are available.

Instances can reside in the same physical or virtual execution environments as long as a replica for a particular primary instance is not in the same server. Redis cluster fails to operate if one of the primary instances and its replicas is unavailable.

The data that is stored in Redis:

- Single-Sign-On sessions
- Access tokens
- Refresh Tokens (for caching)
- Persistent IDs (for caching)

Considerations

Disaster recovery

If disaster recovery is crucial requirement in the deployments where Ubisecure SSO is deployed in high-performance mode, it is suggested to avoid forming a Redis cluster across multiple data centers. This could introduce unnecessary latencies and unpredictability that has not been tested by Ubisecure. Also, in order for the replica election process to work, there must be uneven number of primary instances deployed. Having the same sized Redis clusters in two different data centers would mean that if one data center goes down, the replica election process would not work.

Instead, the two data centers should have their own Redis clusters and the other data center should act as a standby system. In case of failover, the end-users experience a loss of their SSO session and are required to authenticate again.

Migration session storage from LDAP to Redis

When migrating an existing deployment to use Redis as a session storage, it means that existing Single-Sign-On sessions are lost and users will be prompted to authenticate again.