How to install and configure Redis for Identity Server

Complete Redis guide can be found from https://redis.io/

> Redis is a open source in-memory database project that is hosted: https://redis.io/
>
> Officially Redis does not support windows installation and is meant to be installed on Linux platform. Unofficial Windows package exists, but it does not have Redis project support. Therefore this material consists only Linux platform related content.
>
> Redis can be installed even to one instance, however this test setup script installs three Redis instances, without slaves, so that the sharding can be tested. To test High Availability properly, also slave instances are needed.

> **Managing Redis passwords**
>
> In the following examples, we store Redis passwords in a plain text file for the purposes of illustration. However we recommend to use operational best practices when managing system secrets.
>
> Within our examples we use the command line switch "-a" to pass authentication information between Redis CLI and the Redis processes. Using this method will output an expected warning to console.

- Install Redis
  - Redis host and node configuration
  - Open firewall ports to and between instances
  - Install pre-requisites.
  - Download and install Redis in all instances
  - Configure kernel parameters
  - Create configurations
  - Create Redis password
  - Create systemd unit files
  - Enable service scripts
- Start Redis
  - Start Redis in all instances
  - Initialize Redis
  - Redis cluster info
  - Redis cluster nodes
  - Stop Redis
- Configure SSO to use Redis as session storage
  - Add Redis as a session storage

Install Redis

As an administrator install latest stable Redis by compiling it from http://download.redis.io/

- We tested version 5.0.2

Do all these steps in all Redis instances.

Note! IP's can also be Fully Qualified Domain Names if those are available, but for clarity's sake in this document we use IPV4 addresses.

Redis host and node configuration

Minimum high availability and high performance cluster configuration

- This can withstand single host machine going down or single master Redis master node going down.

	Host 1	Host 2	Host 3
Master	A	B	C
Slave	C1	A1	B1

Open firewall ports to and between instances

- Each instance: 7000/tcp, 7001/tcp, 17000/tcp, 17001/tcp

Install pre-requisites.

```
# Become root
sudo su -

# Install pre-requisites
yum -y install gcc make tcl wget

# Create redis user
useradd -m redis

# Create password for user
passwd redis
```

Download and install Redis in all instances

```
# Be root
mkdir -p /usr/local/src/redis
cd /usr/local/src/redis
wget http://download.redis.io/releases/redis-5.0.2.tar.gz
tar xvf redis-5.0.2.tar.gz
cd redis-5.0.2
make
make test
make install
ln -s /usr/local/bin/redis-cli /bin/redis-cli
```

Configure kernel parameters

```
# Be root
sysctl -w net.core.somaxconn=511
sysctl -w vm.overcommit_memory=1


# Persist changes
sysctl -p
```

Create configurations

- Note, if *`hostname -i`* does not give one's externally facing ipv4 address, use the correct value there.
- Add as much **maxmemory** to your Redis as you are capable!
    - As per our findings, you'll get close when calculating ~3,5-5kB per session for maximum 60 minutes of expiry duration.
    - Example: Depending on SSO environment size, we run different kinds of loads during 60 minute time and generated 2.2 million sessions which was divided as follows between the three masters

Master	Peak memory usage
Master 1	`used_memory_peak_human:1.87G`
Master 2	`used_memory_peak_human:1.72G`
Master 3	`used_memory_peak_human:1.70G`

- So by using 1,9G as a good average so the memory requirement for sustained 2,2 million session handling for the cluster per 60 minutes (default configured session expiry duration) per redis host 4.5G would be enough for host machine and master and slave redis instances.
- During our tests, we used also **maxmemory-policy allkeys-lru** (see: https://redis.io/topics/lru-cache) to protect the instances from crashing in case redis would need more memory than was actually available. This however can cause a issue where the session is purged before it expires and throws out errors in SSO because it does not exists anymore.

> ⚠ **Resoving IP of the node**
>
> Note that the following script relies on `hostname -i` to obtain the public IP address of the host. Depending on the setup this might not always give the desired public IP so it is essential that the IP is either manually entered or resolved with a method to works for a specific configuration.

```
# Be root before running these

# Create both Master and Slave configuration
for PORT in 700{0..1}; do

# Create folders
mkdir -p /var/lib/redis/example1-cluster/$PORT /var/log/redis

# Make configuration
cat <<EOF > /var/lib/redis/example1-cluster/$PORT/redis.conf
port $PORT
protected-mode no
cluster-enabled yes
cluster-config-file nodes.conf
cluster-node-timeout 5000
appendonly no
save ""
dir /var/lib/redis/example1-cluster/$PORT
logfile /var/log/redis/redis-cluster-$PORT.log
bind $(hostname -i) 127.0.0.1
maxmemory 3500mb
maxmemory-policy allkeys-lru
EOF

# Own everything created
chown -R redis. /var/lib/redis /var/log/redis
done
```

Create Redis password

> ⚠ Use the same password in all instances!

```
# Be root

# Do this only once, copy to all hosts in master node folder!
# Create strong password, f.ex. with sha256sum
cat /dev/urandom \
  | tr -dc 'a-zA-Z0-9' \
  | fold -w 32 \
  | head -n 1 \
  | sha256sum \
  | awk '{ print $1 }' \
> /var/lib/redis/example1-cluster/7000/redis.password
```

Do this step in all instances including first one!

```
# Be root

echo requirepass $(cat /var/lib/redis/example1-cluster/7000/redis.password) \
  >> /var/lib/redis/example1-cluster/7000/redis.conf
echo requirepass $(cat /var/lib/redis/example1-cluster/7000/redis.password) \
  >> /var/lib/redis/example1-cluster/7001/redis.conf
echo masterauth $(cat /var/lib/redis/example1-cluster/7000/redis.password) \
  >> /var/lib/redis/example1-cluster/7000/redis.conf
echo masterauth $(cat /var/lib/redis/example1-cluster/7000/redis.password) \
  >> /var/lib/redis/example1-cluster/7001/redis.conf
```

Create systemd unit files

```
# Be root

# Enable kernel parameter transparent hugepage via systemd
cat <<EOF > /etc/systemd/system/redis-transparent_hugepage.service
[Unit]
Description=Redis transparent_hugepage

[Service]
Type=simple
ExecStart=/bin/sh -c "echo 'never' > /sys/kernel/mm/transparent_hugepage/enabled && echo 'never' > /sys/kernel
/mm/transparent_hugepage/defrag"

[Install]
WantedBy=multi-user.target
EOF

# Create Redis Master Unit files
cat <<EOF > /etc/systemd/system/redis-cluster-7000.service
[Unit]
Description=Redis persistent key-value database
After=network-online.target redis-transparent_hugepage.service
Wants=network-online.target redis-transparent_hugepage.service

[Service]
ExecStart=/usr/local/bin/redis-server /var/lib/redis/example1-cluster/7000/redis.conf --supervised systemd
ExecStop=/usr/local/bin/redis-cli -h localhost -p 7000 shutdown
Type=notify
User=redis
Group=redis
RuntimeDirectory=redis
RuntimeDirectoryMode=0755
LimitNOFILE=100000 # Recommended minimum 10032 will cause systemwide crash with 100 concurrent user load with
4vcpu SSO in less than 1h

[Install]
WantedBy=multi-user.target
EOF

# Create Redis Slave Unit files
cat <<EOF > /etc/systemd/system/redis-cluster-7001.service
[Unit]
Description=Redis persistent key-value database
After=network-online.target redis-transparent_hugepage.service redis-cluster-7000.service
Wants=network-online.target redis-transparent_hugepage.service redis-cluster-7000.service

[Service]
ExecStart=/usr/local/bin/redis-server /var/lib/redis/example1-cluster/7001/redis.conf --supervised systemd
ExecStop=/usr/local/bin/redis-cli -h localhost -p 7001 shutdown
Type=notify
User=redis
Group=redis
RuntimeDirectory=redis
RuntimeDirectoryMode=0755
LimitNOFILE=100000 # Redis.io recommended minimum 10032 will cause systemwide crash
                   # with 100 concurrent user load with m5.xlarge SSO in less than 1h

[Install]
WantedBy=multi-user.target
EOF
```

Enable service scripts

```
# Be root

# Reload all new unit scripts
systemctl daemon-reload

# Enable newly created Services (so starts after reboot)
systemctl enable redis-transparent_hugepage.service
systemctl enable redis-cluster-7000.service
systemctl enable redis-cluster-7001.service
```

Start Redis

Start Redis in all instances

```
# Be root

# Note, this single command will start all redis services in correct order
systemctl start redis-cluster-7001.service
```

Initialize Redis

- Execute this only in Redis host A towards master node

```
# Be root

# Create helper script for creating cluster
touch /usr/local/bin/create-ids-redis-cluster.sh
chmod +x /usr/local/bin/create-ids-redis-cluster.sh
ln -s /usr/local/bin/create-ids-redis-cluster.sh /usr/bin/create-ids-redis-cluster.sh

cat <<\EOF > /usr/local/bin/create-ids-redis-cluster.sh
#!/bin/bash

# Variables
echo "Please give the ipv4 (10.10.10.10) addresses."
read -p "Redis host 1: " REDIS_HOST1
read -p "Redis host 2: " REDIS_HOST2
read -p "Redis host 3: " REDIS_HOST3
echo
while true; do
    read -p "Are the cluster end-points correct?" yn
    case $yn in
        [Yy]* ) break;;
        [Nn]* ) exit;;
        * ) echo "Please answer yes or no.";;
    esac
done

# Initialize Redis cluster so that master and slave are never in the same host
redis-cli -a $(cat /var/lib/redis/example1-cluster/7000/redis.password) --cluster create -h localhost -p 7000 \
  $REDIS_HOST1:7000 \
  $REDIS_HOST2:7001 \
  $REDIS_HOST2:7000 \
  $REDIS_HOST3:7001 \
  $REDIS_HOST3:7000 \
  $REDIS_HOST1:7001 \
 --cluster-replicas 1

# Distribute redis hash slots between masters
redis-cli -a $(cat /var/lib/redis/example1-cluster/7000/redis.password) -c -h $REDIS_HOST1 -p 7000 cluster
addslots $(for i in {0..5462}; do echo -n " $i"; done)
redis-cli -a $(cat /var/lib/redis/example1-cluster/7000/redis.password) -c -h $REDIS_HOST2 -p 7000 cluster
addslots $(for i in {5463..10924}; do echo -n " $i"; done)
redis-cli -a $(cat /var/lib/redis/example1-cluster/7000/redis.password) -c -h $REDIS_HOST3 -p 7000 cluster
addslots $(for i in {10925..16383}; do echo -n " $i"; done)
EOF

# Execute cluster creation
create-ids-redis-cluster.sh
```

Redis cluster info

```
# Be root
redis-cli -a $(cat /var/lib/redis/example1-cluster/7000/redis.password) -c -h localhost -p 7000 cluster info
```

```
Warning: Using a password with '-a' or '-u' option on the command line interface may not be safe.
cluster_state:ok
cluster_slots_assigned:16384
cluster_slots_ok:16384
cluster_slots_pfail:0
cluster_slots_fail:0
cluster_known_nodes:6
cluster_size:3
cluster_current_epoch:8
cluster_my_epoch:7
cluster_stats_messages_ping_sent:12608
cluster_stats_messages_pong_sent:12569
cluster_stats_messages_sent:25177
cluster_stats_messages_ping_received:12569
cluster_stats_messages_pong_received:12603
cluster_stats_messages_received:25172
```

Redis cluster nodes

```
# Be root
redis-cli -a $(cat /var/lib/redis/example1-cluster/7000/redis.password) -c -h localhost -p 7000 cluster nodes
```

```
Warning: Using a password with '-a' or '-u' option on the command line interface may not be safe.
dc89199dd4799136f2ae88dab6231b4b240250ea xxx.xxx.xx3:7001@17001 slave 22da6c40f786deec295b47289627b3361a50bb7c
0 1545403636000 3 connected
22da6c40f786deec295b47289627b3361a50bb7c xxx.xxx.xx2:7000@17000 master - 0 1545403637000 2 connected 5463-10924
e3cff78f2cb8f93e2916fd0fc6d2c13b38742c2c xxx.xxx.xx3:7000@17000 slave 6ba15487491938cde5dd77cbeddbf01af24e8214
0 1545403636000 8 connected
4c31f7962785f22d9a050cc375bbc2399172bf27 xxx.xxx.xx2:7001@17001 master - 0 1545403637705 7 connected 0-5462
6ba15487491938cde5dd77cbeddbf01af24e8214 xxx.xxx.xx0:7001@17001 master - 0 1545403637504 8 connected 10925-16383
ed296445730f0177f10d37a75b6cdaaa836256ee xxx.xxx.xx0:7000@17000 myself,slave
4c31f7962785f22d9a050cc375bbc2399172bf27 0 1545403637000 0 connected
```

Stop Redis

Stop running Redis processes with this command in all hosts

```
# Be root
systemctl stop redis-cluster-700*
```

Configure SSO to use Redis as session storage

Add Redis as a session storage

To configure SSO to use Redis backed session storage, you need to modify data in SSO configuration database (Ubilogin Directory):

- Create a new ubiloginService entry in cn=Services,ou=System with following attributes:
  - ubiloginClassname *com.ubisecure.ubilogin.session.manager.redis.SessionManagerFactoryRedis*
  - ubiloginConfString url <URL of the Redis service, in form *redis://[address]:[port]*>
    - Use address and port of only one Redis master node. SSO will discover addresses of other master nodes through Redis protocol.
  - ubiloginConfString password <password for the Redis service>
- Link the created ubiloginService to the cn=ServerSession,ou=System entry using ubiloginServiceDN attribute

For example:

```
dn: cn=SessionManagerFactoryRedis,cn=Services,ou=System,@suffix@
changetype:add
objectClass: ubiloginService
cn: SessionManagerFactoryRedis
ubiloginClassName: com.ubisecure.ubilogin.session.manager.redis.SessionManagerFactoryRedis
ubiloginConfString: url redis://redisnode1.example.com:7000
ubiloginConfString: password SecretPassword1

dn: cn=ServerSession,ou=System,@suffix@
changetype: modify
replace: ubiloginServiceDN
ubiloginServiceDN: cn=SessionManagerFactoryRedis,cn=Services,ou=System,@suffix@
-
```

Note that `@suffix@` must be expanded to the value of attribute *suffix* in unix.config.

The change can be done using for example Apache DirectoryStudio, or you can create an ldif file to change the file, and load the data using import script:

```
./ldap/openldap/import.sh ldap/[name of file containing the changes].ldif
```

Restart required!

After this this change, please restart all SSO services so that the change becomes active.

Additional failover addresses

Ubisecure SSO only accepts one address to connect to Redis. However, if the single Redis instance is reachable when SSO starts, it will then receive all Redis instances and there is no single point of failure after initial connection is established.

Remove Redis as a session storage

To return SSO to use the default session storage UbiloginDirectory, you only need to delete the ubiloginServiceDN attribute from the cn=ServerSession, ou=System entry.

```
dn: cn=ServerSession,ou=System,@suffix@
changetype: modify
delete: ubiloginServiceDN
-
```

The change can be done using for example Apache DirectoryStudio, or you can create an ldif file to change the file, and load the data using import script:

```
./ldap/openldap/import.sh ldap/[name of file containing the changes].ldif
```