Using the High Availability (HA) Toolkit

On this page:

The AppDynamics HA Toolkit helps you set up and administer a high availability (HA) deployment of AppDynamics Controllers. This topic describes how to use the toolkit to manage Controllers as a high availability pair.

About the HA Toolkit

The HA toolkit consists of bash scripts that automate HA-related set up and administration tasks for the Linux operating system. It works with most flavors of Linux, including Ubuntu and Red Hat/CentOS.

You can use the toolkit to:

- Configure Controllers in a high availability pair arrangement.
- Install a watchdog process on the secondary that monitors the primary Controller and automatically fails over to the secondary when needed.
- Install the Controller as a Linux service.
- Fail over to a secondary Controller manually (for example, when when you need to perform maintenance on the primary).
- Revive a Controller (restore a Controller as an HA secondary after its database is more than seven days behind the master as a replication slave).

Deploying Controllers as an HA pair ensures that service downtime in the event of a Controller machine failure is minimized. It also facilitates other administrative tasks, such as backing up data. For more background information, including the benefits of HA, see Controller High Availability (HA).

The toolkit works on Linux systems only. However, even if you cannot use the HA toolkit directly (due to a different operating system or because of site-specific requirements), you are likely to benefit from learning how the toolkit works since it can provide a model for your own scripts and processes.

You can download the HA toolkit from the following GitHub link: https://github.com/Appdynamics/HA-toolkit/releases.

Before Starting

In an HA deployment, the HA toolkit on one Controller host needs to be able to interact with the other Controller host. The toolkit relies on certain conditions in the environment to support this interaction, along with the other operations it performs.

General guidelines and requirements for the environment in which you deploy HA are:

- Two dedicated machines running Linux. The Linux operating systems can be Fedora-based Linux distributions (such as Red Hat or CentOS) or Debian-based Linux distributions (such as Ubuntu).
- One of the machines should have the Controller installed. The second machine should not have a Controller installed, as the HA Toolkit replicates the Controller from the primary machine to the secondary.

- In a Controller HA pair, a load balancer should route traffic from Controller clients (Controller UI users and App Agents) to the active Controller. Before starting, make sure that a load balancer is available in your environment and that you have the virtual IP for the Controller pair as presented by the load balancer.
- Port number 3388 should be open between the machines in an HA pair.
- The login shell must be bash (/bin/bash).
- The host machines should have identical directory structures.
- A network link connecting the HA hosts can support a high volume of data. The primary and secondary may be in separate data centers, but a dedicated network link between the hosts is recommended.
- SSH keys on each host that allows SSH and rsync operations by the AppDynamics user (instructions below).
- The hosts file (/etc/hosts) on both Controller machines should contain entries to support reverse lookups for the other node in the HA pair (instructions below).
- Because Controller licenses are bound to the network MAC address of the host machine, the HA secondary Controller will need an additional HA license. You should request a secondary license for HA purposes in advance.
- The location of the Controller home directories needs to be writable by the user under which the Controller runs.
- Super user access on the machine to install system services on the host machine, either as root or as a user in the sudoers group. See the next section for considerations and additional requirements for running as a non-root (but sudo-capable) user.

User Privilege Escalation Requirements

If the AppDynamics Controller is run by a non-root user on the system, the HA toolkit process must be able to escalate its privilege level to accomplish certain tasks, including replication, failover and assassin tasks.

The script uses one of the following two mechanisms to accomplish privilege escalation. The toolkit installation adds artifacts required for both mechanisms; however, the one that it actually uses is determined at runtime based on dependencies in the environment, as follows:

- /etc/sudoers.d/appdynamics contains entries to allow the AppDynamics user to access the /sbin/service utility using sudo without a password. This mechanism is not available if the AppDynamics user is authenticated by LDAP.
- /sbin/appdservice is a setuid root program distributed in source form in HA/appdservice.c. It is written explicitly to support auditing by security audit systems. The install-init.sh script compiles and installs the program. It is executable only by the AppDynamics user and the root user. The script requires a C compiler to be available on the system. You can install a C compiler using the package manager for your operating system. For example, on Yum-based Linux distributions, you can use the following command to install the GNU Compiler, which includes a C compiler:

```
sudo yum install gcc
```

After deploying the HA Controller pair, you will be able to test support for one of these functions (without causing system changes) by running these commands as the appd user:

- /sbin/appdservice appdcontroller status
- sudo /sbin/service appdcontroller status

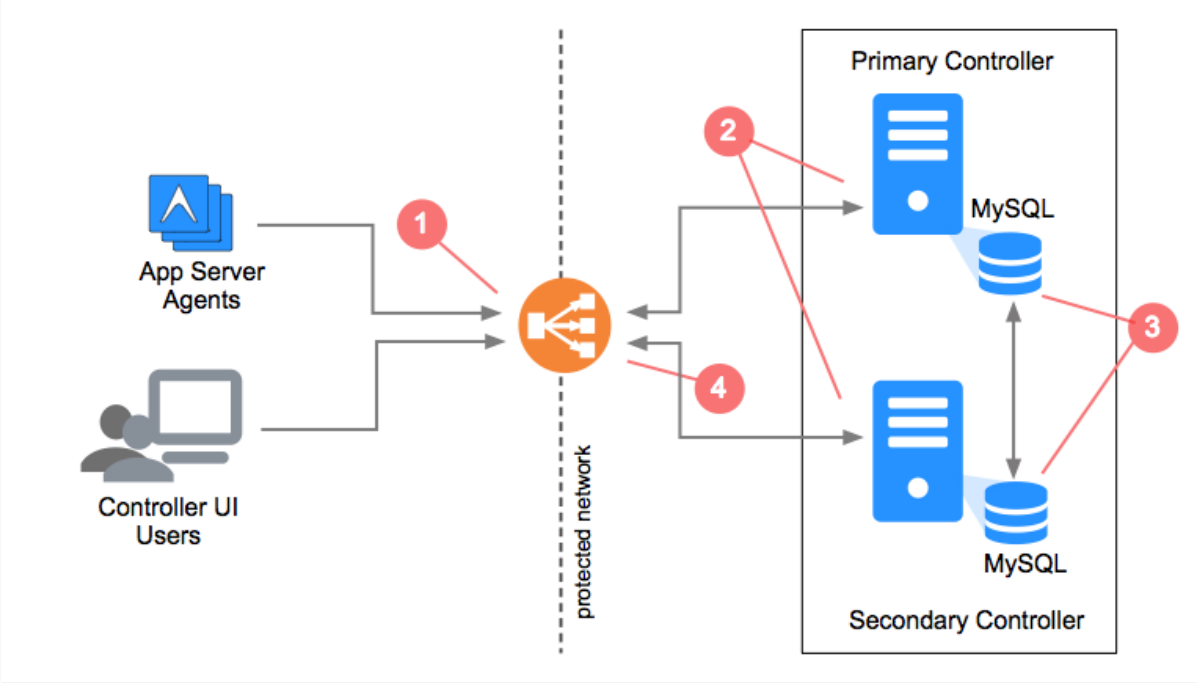
At least one of the commands must successfully return the Controller status for the toolkit to work.

Load Balancer Requirements and Considerations

Before setting up HA, a reverse proxy or load balancer needs to be available and configured to route traffic to the active Controller in the HA pair. Using a load balancer to route traffic between Controllers (rather than other approaches, such as DNS manipulation) ensures that a failover can occur quickly, without, for example, delays due to DNS caching on the agent machines.

An HA deployment requires the following IP addresses:

- One for the virtual IP of the Controller pair as presented by the load balancer used by clients, such as App Agents, to access the Controller. (Callout 1 in the following diagram.)
- Two IP addresses for each Controller machine, one for the HTTP primary port interface (2) and one for the dedicated link interface between the Controllers (3) on each machine. The dedicated link is recommended but not mandatory.
- If the Controllers will reside within a protected, internal network behind the load balancer, you also need an additional internal virtual IP for the Controller within the internal network (4).



When configuring replication, you specify the external address at which Controller clients, such as app agents and UI users, will address the Controller at the load balancer. The Controllers themselves need to be able to reach this address as well. If the Controller will reside within a protected network relative to the load balancer, preventing them from reaching this address, there needs to be an internal VIP on the protected side that proxies the active Controller from within the network. This is specified using the -i parameter.

The load balancer can check the availability of the Controller at the following address:

```
http://<controller_host>:<port>/controller/rest/serverstatus
```

If the Controller is active, it responds to a GET request at this URL with an HTTP 200 response. The body of the response indicates the status of the Controller in the following manner:

```
<serverstatus vendorid="" version="1">
   ...
   <available>true</available>
   ...
```

Ensure that the load balancer policy you configure for the Controller pair can send traffic to only a single Controller in the pair at a time (i.e., do not use round-robin or similar routing distribution policy at the load balancer). For more information about setting up a load balancer for the Controller, see Use a Reverse Proxy.

Setting Up the Controller High Availability Pair

Setting up high availability involves the following steps:

- Step 1: Configure the Environment
- Step 2: Install the HA Toolkit
- Step 3: Prepare the Primary Controller
- Step 4: Set Up the Secondary Controller and Initiate Replication

Step 1: Configure the Environment

The following sections provide more information on how to configure a few of the system requirements. They describe how to configure the settings on Red Hat Linux for a sample deployment. Note that the specific steps for configuring these requirements may differ on different systems. Consult documentation for your system for details on that system.

Host Reverse Lookups

Reliable symmetrical reverse host lookup needs to be set up on each machine. The best way to accomplish this is by placing the host names of the pair into the hosts files (/etc/hosts) on each machine. This is preferable over other approaches, namely using reverse DNS, which adds a point of failure.

To enable reverse host lookups, on each host:

1. In /etc/nsswitch.conf, put "files" before "dns" to have the hosts file entries take precedence over DNS. For example:
   hosts: files dns
2. In /etc/hosts file, add an entry for each host in the HA pair, as in the following example:
   192.168.144.128 host1.domain.com host1
   192.168.144.137 host2.domain.com host2

> It is important to adhere to the correct format of /etc/hosts files in order to reduce errors. If you have both dotted hostnames and short versions, you need to list the dotted hostnames with the most dots first and the other versions subsequently. This should be done consistently for both HA server entries in each of the two /etc/hosts files. Note in the examples above that the aliases are listed last.

Set up the SSH key

SSH must be installed on both hosts in a way that gives the user who runs the Controller passwordless SSH access to the other Controller system in the HA pair. You can accomplish this by generating a key pair on each node, and placing the public key of the other Controller into the authorized keys (authorized_keys) file on each Controller.

The following steps illustrate how to perform this configuration. The instructions assume an AppDynamics user named appduser, and the Controller hostnames are node1, the active primary, and node2, the secondary. Adjust the instructions for your particular environment. Also note that you may not need to perform every steps (for example, you may already have the .ssh directory and don't need to create a new one).

Although not shown here, some of the steps may prompt you for a password.

On the primary (node1):

1. Change to the AppDynamics user, appduser in our example:

```
su - appduser
```

2. Create a directory for SSH artifacts (if it doesn't already exist) and set permissions on the directory, as follows:

```
mkdir -p .ssh
chmod 700 .ssh
```

3. Generate the RSA-formatted key:

```
ssh-keygen -t rsa -N "" -f .ssh/id_rsa
```

4. Secure copy the key to the other Controller:

```
scp .ssh/id_rsa.pub node2:/tmp
```

On the secondary (node2):

1. As you did for node1, run these commands:

```
su - appduser
mkdir -p .ssh
chmod 700 .ssh
ssh-keygen -t rsa -N "" -f .ssh/id_rsa
scp .ssh/id_rsa.pub node1:/tmp
```

2. Add the public key of node1 that you previously copied to the secondary Controller host's authorized keys and set permissions on the authorized keys file:

```
cat /tmp/id_rsa.pub >> .ssh/authorized_keys
chmod 700 ~/.ssh/authorized_keys
```

On the primary (node1) again:

1. Move the secondary's public key to the authorized keys

```
cat /tmp/id_rsa.pub >> ~/.ssh/authorized_keys
chmod 700 ~/.ssh/authorized_keys
```

To test the configuration, try this:

```
ssh -oNumberOfPasswordPrompts=0 <other_node> "echo success"
```

Make sure the echo command succeeds.

Step 2: Install the HA Toolkit

With your environment configured, you can get and install the HA Toolkit. The toolkit is packaged as a shar (shared archive) file, which, when executed, extracts the set of scripts of the toolkit.

On the primary:

1. Change to the Controller home directory:

```
cd /<path_to_AppD>/AppDynamics/Controller
```

2. Create the HA directory in the Controller home:

```
mkdir -p HA
```

3. Make the entire directory writeable and `cd` into the directory:

```
chmod +w HA
cd HA
```

4. Download the HA toolkit distribution file to the HA directory.
5. Make the file executable:

```
chmod 775 HA.shar
```

6. Run the shell archive script:

```
./HA.shar
```

Step 3: Prepare the Primary Controller

Once you have set up the environment and downloaded the toolkit to the primary Controller, you can set up the primary Controller for high availability. The steps for setting up HA differ if you are deploying a Controller for the first time or adding an HA secondary to an existing Controller, i.e., one that has already accumulated application data:

- To deploy two new Controllers as an HA pair, you first need to install the Controller on the primary machine. For instructions on installing, see Install the Controller with the Platform Admnistration Application or Install the Controller with the Package Installer.
- To add an HA secondary to an existing standalone Controller deployment, you only need to verify that the user who runs the Controller has write access to the Controller home, as described below.

Once installation is finished, ensure that the Controller home and data directories are writable by the AppDynamics user.

Use the `ls` command to verify write privileges. The output should look similar to the output below, which shows the current privileges for the sample AppDynamics user, appduser.

```
ls -lad /opt/AppDynamics/Controller
drwxr-xr-x. 18 appduser users 4096 Jan 26 18:18 /opt/AppDynamics/Controller
```

After preparing the primary, you can set up replication, as described next.

Step 4: Set Up the Secondary Controller and Initiate Replication

To set up the secondary Controller, you run the replicate.sh script on the primary Controller machine. This script is the primary entry point for the toolkit. It performs these functions, among others:

- Deploys the secondary Controller
- Replicates data to the secondary
- Configures master-master data replication between the primary and secondary databases
- Optionally, starts the watchdog process on the secondary that watches the primary and initiates a failover if needed.

This script can only be run on the primary Controller. If you run the replicate script with super user (sudo) privileges, it performs the complete HA setup—from installing the secondary Controller, copying data to the secondary, and setting up master-master database replication. If you do not run the script as a super user, you will need to perform some additional configuration tasks later to install system services. To perform those tasks, run the `Install-init.sh` script as described in Installing as a Service.

You will need to run replicate script at least twice. On the first pass, the script performs initial setup and replication tasks. The final replication pass (which is specified using the -f flag) completes replication and restarts the Controller. This results in a brief service downtime.

For an existing Controller deployment with a large amount of data to replicate, you may wish to execute replication multiple times before finalizing replication. The first time you run the script, it can take a significant amount of time to complete, possibly days. The second pass replicates the data changes accumulated while the first pass executed. If performed immediately after the first pass, the subsequent pass should take considerably less time. You can run the replication pass again until the amount of time it takes for the replicate script to complete fits within what would be an acceptable downtime window.

To set up secondary replication:

1. For the initial replication steps, invoke the replicate script, passing the hostname of the secondary and virtual IP for the Controller pair at the load balancer when invoking the replicate script. The command should be in the following form:

   ```
   ./replicate.sh -s <secondary_hostname> -e <external_vip> -i <internal_vip>
   ```

   Command options are:

   - -s – The hostname of the secondary, as you have configured in /etc/hosts
   - -e *<external_vip>* – The hostname and port for the Controller pair as exposed to clients at the load balancer or other type of reverse proxy. This is the address at which the App Agents and the Controller UI clients would typically address the Controller. For example: http://controllervip.example.com:80.
   - -i *<internal_vip>* – If your Controllers reside within a protected network (relative to the load balancer) and cannot address the virtual IP directly, use this value to indicate an internal virtual IP address for the Controllers to use. For example: http://controllervip.internal.example.com:80.

   For all available options to the script, run the replicate script with no arguments.

   If running as non-root, the command asks that you run the install-init script manually as root to complete the installation.

2. If replicating a large amount of data, repeat the previous command to minimize the time required for the finalization pass in which the Controller is restarted.
3. When ready to finalize replication, run the script again, this time passing the -w and -f flags:

   ```
   ./replicate.sh -s <secondary_hostname> -f -w  -e <external_vip> -i <internal_vip>
   ```

   The flags have the following effect

   -w – Starts the watchdog process on the secondary (see below for more information about the watchdog)

   -f – Causes the script to finalize setup, restarting the Controllers.

4. When finished, the databases on both hosts will be up and replicating, with the primary Controller available for receiving application data and servicing user access. Verify on the secondary Controller that data has been replicated correctly, as follows:
   a. At the command line on the secondary controller, go to the <controller_home>/bin directory.
   b. Log in to the secondary Controller database:

      ```
      controller.sh login-db
      ```

   c. Execute following command:

      ```
      SHOW SLAVE STATUS\G
      ```

      This step should provide you following result:

      ```
      Seconds_Behind_Master: $Number_Of_Seconds_Behind_Master
      ```

```
```

If you get a non-zero number for this test, wait until the number becomes zero.

5. Check the status of the Controller using the commands described in User Privilege Escalation Requirements.

Starting the Controller Availability Watchdog

The watchdog process is a background process that runs on the secondary. It monitors the availability of the primary Controller, and, if it detects that the primary is unavailable, automatically initiates a failover. If you passed the -w flag to the replicate script when setting up HA, the watchdog was started for you. You can start the watchdog manually and configure its default settings as described here.

The watchdog sets a health check every 10 seconds. You can configure how long the watchdog waits after detecting the primary is down before it considers it a failure and initiates failover. By default, it waits 5 minutes, rechecking for availability every 10 seconds. Since the watchdog should not take over for a primary while it is in the process of shutting down or starting up, there are individual wait times for these operations as well.

To enable the watchdog:

1. Copy watchdog.settings.template to a file named watchdog.settings.
2. Edit watchdog.settings and configure the limits settings in the file. In effect, these limits define the thresholds which, when exceeded, trigger failover. The settings are:
    a. `DOWNLIMIT`: Length of time that the primary is detected as unavailable before the watchdog on the secondary initiates failover.
    b. `FALLINGLIMIT`: Length of time that the primary reports itself as shutting down before the watchdog on the secondary initiates failover. The secondary needs to allow the primary to shut down without initiating failover, so this setting specifies the length of time after which the primary may be considered "stuck" in that state, at which point the secondary takes over.
    c. `RISINGLIMIT`: Length of time that the primary reports itself as starting up before the watchdog on the secondary initiates failover. The secondary needs to allow the primary to start up without initiating failover, so this setting specifies the length of time after which the primary may be considered "stuck" in that state, at which point the secondary takes over.
    d. `DBDOWNLIMIT`: Length of time that the primary database is detected as unavailable before the watchdog on the secondary initiates failover.
    e. `PINGLIMIT`: Length of time ping attempts of the primary fail before the secondary initiates failover.
3. Create the control file that enables the watchdog. For example:

```
touch <controller_root>/HA/WATCHDOG_ENABLE
```

4. Enable read/write permissions on the file:

```
chmod 777 <controller_root>/HA/WATCHDOG_ENABLE
```

5. Start the service:

```
service appdcontroller start
```

Note: Running the replicate.sh script with the -w option at final activation creates the watchdog control file automatically.

Removing the WATCHDOG_ENABLE file causes the watchdog to exit.

The Controllers are now configured for high availability. The following sections describe how to perform various administrative tasks in a high availability environment.

Bouncing the Primary Controller without Triggering Failover

To stop and start the primary Controller without initiating failover, remove the watchdog file on the secondary before stopping or initiating the restart on the primary. This causes the secondary to stop watching the primary, so that it doesn't initiate failover when the primary is briefly unavailable.

When the primary is finished restarting, you can add the file back to resume the watchdog process. The file is:

<secondary_controller_home>/HA/WATCHDOG_ENABLE

Starting and Stopping the Controller

After you have set up HA, the Controller is automatically started at boot time and shut down when the system is halted. You can start and stop the Controller service and HA facility manually at any time using the Linux service command as root user.

To start or stop the Controller manually, use the following commands:

- To start:

```
service appdcontroller start
```

- To stop:

```
service appdcontroller stop
```

Installing as a Service

The replicate script installs the Controller as a service for you automatically if you run the script as a root user. If you did not run the replicate script as root, after the replicate script process is finished, you can run the following script manually to complete the installation:

1. Change directories into `<controller_home>/HA`.
2. Run `install-init.sh` with one of the following options to choose how to elevate the user privilege:
    - `-c` #use setuid c wrapper
    - `-s` #use sudo
    - `-p` #use prune wrapper
    - `-x` #use user privilege wrapper

    Optionally, you can also specify the following option: `-a <Machine Agent install directory>`. This option configures the machine agent to report to the Controller's self-monitoring account and install an init script for it.

If you need to uninstall the service later later, use the `uninstall-init.sh` script.

Once installed as a service, the Linux service utility can be run on either node to report the current state of the replication, background processes, and the Controller itself.

To check its status, use this command:

```
service appdcontroller status
```

The toolkit also writes status and progress logs of its various components to the logs.

Performing a Manual Failover and Failback

To fail over from the primary to the secondary manually, run the `failover.sh` script on the secondary. This kills the watchdog process, starts the app server on the secondary, and makes the database on the secondary the replication master. If you have custom procedures you want to add to the failover process (such as updating a dynamic DNS service or notifying a load balancer or proxy), you can add or call it from this script.

Should the secondary be unable to reach the MySQL database on the primary, it will then try to kill the app server process on the primary Controller, avoiding the possibility of having two Controllers active at the same time. This function is performed by the `assassin.sh` script, which continues to watch for the former primary Controller process to ensure that there aren't two active, replicating Controllers.

The process for performing a failback to the old primary is the same as failing over to the secondary. Simply run failover.sh on the machine of the Controller to restore it as the primary. Note that if it has been down for more than seven days, you need to revive the database, as described in the following section.

Reviving a Controller database

The Controller databases can be synchronized using the replicate script if they have been out of sync for more than seven days. Synchronizing a database that is more than seven days behind a master is considered reviving a database. Reviving a database involves essentially the same procedure as adding a new secondary Controller to an existing production Controller, as described in Set Up the Secondary Controller and Initiate Replication.

In short, you run the `replicate.sh` without the `-f` switch multiple times on the primary. Once you have an opportunity for a service window and reduced the replication time to an acceptable amount of time for a service window, take the primary Controller out of service (stop the app server) and allow data synchronization to catch up.

Backing Up Controller Data in an HA Pair

An HA deployment makes backing up Controller data relatively straightforward, since the secondary Controller offers a complete set of production data on which you can perform a cold backup without disrupting the primary Controller service.

After setting up HA, perform a back up by stopping the `appdcontroller` service on the secondary and performing a file-level copy of the AppDynamics home directory (i.e., a cold backup). When finished, simply restart the service. The secondary will then catch up its data to the primary.

Updating the Configuration in an HA Pair

When you run the replicate script, the toolkit copies any file-level configuration customizations made on the primary Controller to the backup, such as configuration changes in domain.xml file.

Over time, you may need to make modifications to the Controller configuration. After you do so, you can use the `-j` switch to replicate configuration changes only from the primary to the secondary. For example:

```
./replicate.sh -j
```

Troubleshooting HA

The HA toolkit writes log messages to the log files located in the same directory as other Controller logs, <controller_home>/logs by default. The files include:

- replicate.log: On the primary machine, this log contains events related to replication and HA setup.
- watchdog.log: On the secondary Controller host, this log contains event information generated by the watchdog process.
- assassin.log: On the secondary machine (or a machine on which the watchdog process has attempted to terminate the remote Controller process) information generated for the attempt to terminate the remote Controller process typically due to a failover event.
- failover.log: Failover event information.