# Upgrading In-House Systems

The typical sequence for upgrading Agiloft is to:

1. Create a backup
2. Perform the upgrade

It is very rare that problems arise during an upgrade, but if they do, one can always roll back to the backup. For moderately sized KBs, the process is simple because the Agiloft installer includes an option to create a full backup and automatically executes a set of integrity checks after the upgrade. Then, if there are any issues, the installer can automatically roll back. The entire process can take less than 15 minutes and a handful of mouse clicks to complete.

# Upgrading Large KBs

With large KBs, the upgrade itself is still fast and typically takes less than an hour, even when the KB includes terabytes of data. However, creating the backup can take a long time if the native Agiloft facility is used because it has to read the database using SQL queries and export it.

The solution is to use native OS and database backup facilities. For example, the Agiloft hosting service uses CephFS snapshots to create a backup of the file system. Because these snapshots use Copy on Write technology, creating the backup only takes a minute or so, even when there are terabytes of data. This is implemented outside of Agiloft because the application does not have access to these native OS-level facilities.

At customer sites, the infrastructure varies widely. Agiloft may be hosted on a bare metal of virtual infrastructure, running against Linux and MySQL, or Windows and SQL Server. In some cases, the database is replicated across multiple servers; in others, the database is on the same machine as the application server. And the file system may be native or SAN-based. However, the basic process for upgrading large KBs is still the same:

1. Create a backup*, using native database facilities to create a backup of the database and native OS-level facilities to create a backup of the filesystem*
2. Perform the upgrade, *with options selected not to create an OS or KB level backup*

Or, you may use the following process with additional steps indicated in blue. These steps allow you to manually ensure that the backups are good and that the production upgrade goes smoothly.

1. Create a backup*, using native database facilities to create a backup of the database and native OS-level facilities to create a backup of the filesystem*
2. Restore the backup onto a server with firewall settings that prevent it from connecting to other systems or sending email
3. Upgrade the backup, *with options selected not to create an OS or KB level backup*
4. Login to the upgraded backup and confirm that everything is working properly
5. Create a new backup of the original Agiloft instance*, using native database facilities to create a backup of the database and native OS-level facilities to create a backup of the filesystem*
6. Perform the upgrade, *with options selected not to create an OS or KB level backup*

The additional advantages of this process is that steps 2 through 4 can be executed at any time without interfering with production use, and since the backup is a copy of the original KB, any upgrade issues will be found when upgrading the backup.

In order to assist in-house customers with the filesystem backup, we have documented the Agiloft directory structure here.

⚠ For customers who want to deploy in-house without dealing with the complexity of using native database facilities to create a backup of the database, we recommend installing on Linux using the default installation parameters.

This results in MySQL being installed on the same machine as the WildFly application server, so the installer can create a backup of the file system and database by simply shutting down the database and creating a .tar file. This takes longer than a snapshot, but much less time than a KB-level export. Having the database and application server on the same machine also significantly improves performance. Security is also maintained because the database is automatically configured to only listen to connections from localhost.