Server updates - OS updates

To ensure secure operation, servers in the infrastructure have to be kept up to date. This tutorial addresses the need for OS updates of the IdM server and gives basic guidelines and recommendations.

Update strategy

Each organization has some sort of schedule to apply OS patches: weekly, monthly, quarterly, never (not a good one), etc. You can patch the OS according to your strategy, but we recommend to perform patching at least once every three months. IdM relies on packages and libraries from the operating system and if those are not patched, security of the whole IdM solution also deteriorates.

Things to consider

Before applying updates, there are few things to consider:

- Impact on users
    - IdM is often deployed as a self-service portal for users. You should plan the downtime such that minimal number of users is affected.
    - Users may make changes in the IdM that start some long running tasks (e.g. automatic roles changes, bulk role assignments, etc.). Those tasks are executed asynchronously and may be running even if the user who started the task has already logged off.
- Impact on long running tasks (LRT)
    - IdM has internal cron that schedules LRT jobs. To make things safe, no job should be running when you are doing the update. The safest way to achieve this is to stop the IdM service before applying updates.
    - LRTs run usually at night so it is not entirely necessary to stop the IdM, but you have to make sure you have enough time to perform the patching (and possible rollback) before jobs start to execute.
    - Restarting IdM cancels the LRT that was currently running, LRT **will not pick up automatically** after IdM goes up again.
    - Nightly LRTs usually read HR system data. This means there are dependecies between them (e.g. synchronize identities, then contracts and/or time slices, then run recompute on them and finally run HR processes which enable/disable identities based on freshly synchronized data). Given the nature of deployment, those dependencies may be "hard" and it may be dangerous to skip some of LRTs or run them in different order.
- Impact on entity events
    - Entity events that are currently running **are lost** on IdM restart. This usually affects from one to ten events; actual number of affected events depends on number of `event-executor` threads.
    - Entity events in other states are persisted into the database so they are not lost on IdM restart.
    - No entity events should be in the event queue at the time of OS update. Because events are generated by LRTs or user actions, killing off LRTs and disconnecting users from IdM

web interface is sufficient.
- Impact on end systems connected to IdM
    - There is no direct impact on other systems.
    - There **may be** some impact on connections the IdM makes in order to manage end systems.
        - Some end systems use SSL-secured form of communication and IdM needs to have their certificate in its Java truststore. If the truststore was improperly configured and gets regenerated (i.e. due to ``ca-certificates`` package update), all extra certificates are lost making SSL connections to end systems fail. This should not happen because IdM should use its own, explicitly created and configured, truststore.
        - Some end systems that are connected via WinRM. The WinRM library uses Python and some of Python's libraries come from the OS packages. Upgrading those packages system-wide has **possibly** an impact on the way the WinRM/Python works.
- Impact on OS
    - OS may seemingly not boot after the updates (boot or network issues, SSHd/RDP daemon issues). We recommend to have complete backup of ``/boot`` and ``/etc`` directories. Out-of-band access to a machine is a must. In case of virtualized environment, making a snapshot is a way to go.
    - In our deployments, we use mainly RHEL/CentOS (sometimes Debian) and Windows OSes. If you deploy IdM accordingly (tutorials here and here), OS updates are generally painless.
    - Packages from OS that IdM deployment uses
        - Java (java binary referenced through ``/usr/lib/…`` and therefore through ``/etc/alternatives/…``). Java patchset may be updated, but the version should stay the same (e.g. update ``1.8u27→1.8u90`` is OK, but update ``Java8→Java9`` is not).
        - PostgreSQL is installed generally from OS or PGDG repositories and is considered pretty stable. Updating package when PostgreSQL version stays the same is OK. Updating PostgreSQL version (e.g. ``9.6→10``) should be OK, but we recommend at least to make a backup of IdM database (in case you have to rollback the previous PostgreSQL version).
        - Apache HTTPD. Deployment should be stable and no special care is needed. We recommend to have a backup of vhost configuration.
    - Windows-based installations have all deployment components installed by-hand and therefore are not really susceptible to break by OS updates. But this also means you have to update all deployment components manually.
- Finding bugs
    - It is for the best to have at least two environments - test env. and production env.
    - Update the test environment first, then leave it running for at least one week. If no bugs are found by then, you can update the production environment. The one week provides minimal safe time frame where some of the bugs can manifest (e.g. memleaks).
    - Define use-cases that are important for your deployment. Before and after the update, test if those use-cases work.

Performing the OS update

Following list can be used as a basis for the maintenance checklist. Feel free to customize it to better suit your needs.

1. Preparations
    1. Prepare testing use-cases.
    2. Prepare backup and restore procedures.
    3. Identify which LRTs can be safely killed when running.
    4. Make a checklist with timing estimates to determine the length of the maintenance.
2. Perform the update
    1. Begin the maintenance.
    2. Disable monitoring system notifications.
    3. (If you use hot snapshots, make one.)
    4. Make sure no user or external application can access the IdM.
    5. Log into the IdM as administrator and check if there are some LRTs running.
        1. If they are not, continue.
        2. If they are, either stop those LRTs or let them finish. This depends on your deployment.
    6. Stop the IdM.
    7. Disable automatic start of the IdM on OS start.
    8. (If you use cold snapshots, turn off the machine and make one.)
    9. (If you do not use snapshots, make a backup of the IdM database and store it off-machine.)
    10. Make backup of ``/boot``, ``/etc``, list of processes ``ps -ef`` and list of network services ``netstat -tulnp`` (or ``ss -tulnp``). Those dumps will help you check if all the services started. You can also recover some settings from backups in case something goes wrong (in a minor way) - you will not need to roll back whole snapshot.
    11. Perform the update (e.g. ``yum update``).
        1. YMMV depending on the packages being updated. Also when upgrading PostgreSQL, there are additional steps you have to perform.
    12. Restart affected services or reboot the whole machine if necessary.
    13. When the machine is up, check ``dmesg`` and ``/var/log/{messages,syslog}`` or analogous files for your OS.
    14. Check running processes and network services whether everything started properly.
        1. Namely PostgreSQL and HTTPd should be up and running. Those are parts of IdM deployment.
    15. If everything is ok, start the IdM service.
    16. Check IdM logs whether it started successfuly.
    17. Log into the IdM and test connection to end systems (configuration form for the system, green button "Test connector").
    18. Check your testing use-cases.
    19. Enable autostart of IdM service upon OS start.
    20. (If there were changes to the database (e.g. PostgreSQL major version upgrade), make a backup of the upgraded database.)
    21. Allow users to access the IdM.
    22. Enable monitoring system notifications.
    23. End the maintenance.
3. Wrap-up
    1. Update documentation if necessary.
    2. Perform maintenance analysis and update your procedures if necessary.
    3. Update your test cases if necessary.
    4. After about a week, check system logs to make sure all components work as expected.

For Windows OSes, the update process is roughly the same. For checking services, status of the system and system logs, use the Event Viewer and Server Manager.

Resolving issues

For maintenance actions, it is necessary to:

- Know how long each task will take and to measure the task duration when actually performing them.
  - If tasks take longer than expected, you know if you can match the maintenance window or not.
- Know how long the whole maintenance will take (maintenance time **MT**).
  - This is not simply a sum of task times, you should add some extra time (**ET**) to have a proper cushion.
- Know how long (at worst) the whole rollback will take (rollback time **RT**).
- Have a maintenance window that spans at least **MT**+**RT** with some extra time **ET**.
  - You are not able to safely perform the maintenance in shorter window, there is simply not enough time. If something goes wrong, you will need **RT** time to perform the rollback!
  - When you have no **ET**, if anything goes wrong you have to perform rollback procedure. Therefore, **ET** gives you some time you can spend on solving the issue so you can carry on with updates.

- You should have a rollback procedure that can safely restore the deployment.
  - This depends on your environment and on the way you updated OS packages.
- Fortunately, in most cases it simply means restoring the snapshot of the virtual machine.
  - After restoring the snapshot, you have to perform tests (with test use-cases) to confirm the rollback was performed correctly.
  - Minor issues can be generally resolved with the help of ``/boot`` and ``/etc`` backups you created before updating the OS.
- If IdM installation gets hit, you can debug the configuration or restore it from periodic backup. Since IdM is not installed from OS packages, this basically never happens.