Paper 1825-2018

Taming Change: Bulk Upgrading SAS® 9.4 Environments to a New Maintenance Release

Javor Evstatiev, Andrey Turlov

ABSTRACT

The deployment options for the SAS® 9 Platform have changed dramatically in the last years. Cloud based deployments are becoming a real alternative. On premise deployments are facing pressure to reduce operational costs. Downtimes are expected to be short to allow end users an almost seamless usage of the platform. Some organizations maintain several SAS® installations (deployments) in multiple stages for separate business entities because of different regulatory, technical and business requirements.

A widely used approach is to hotfix SAS® environments only for “alert” level issues and introduce new functionality with maintenance level upgrades. Maintenance level upgrades are far more complex than deploying a hotfix, but they are smaller in scope, visibility, and complexity than a release upgrade. In the following paper we discuss strategies for maintenance level upgrades and how to successfully apply them to multiple SAS® deployments.

INTRODUCTION

Why should we upgrade at all? Drivers for upgrading may be:

- Business requires it (to make use of new features or receive fixes to issues)
- Technical (security hotfixes, major infrastructure changes, vulnerability changes)
- Compliance (for example introduction of TLS 1.2 companywide)
- Service Level Agreements (towards end users of the platform)

In addition, the costs of not upgrading must be taken into consideration. For instance, any one of the following items may result in massive increases in operational costs:

- Remaining the last and only application that needs the legacy version of a database
- Running out of vendor support
- Increased downtimes and service disruptions
- Non-compliance with new regulations or governance practices (think GDPR)

Some terms used on the following pages may mean different things to different people. Here a brief introduction to the terms as we understand them:

- User generated content is any type of content created by your end users. This may be Enterprise Guide projects, Base SAS® code and macros, Data Integration Studio Jobs, Stored Processes, Enterprise Miner Projects and similar.
- The SAS Platform uses an n-tier architecture to distribute functionality across resources. Typical tiers are Middle tier, Server tier, Data tier and Client tier.
- Environment is a single deployment (installation) of the SAS® Platform, spanning multiple tiers

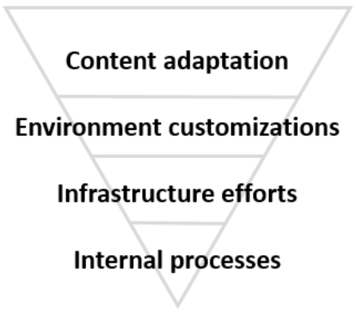
and servers.

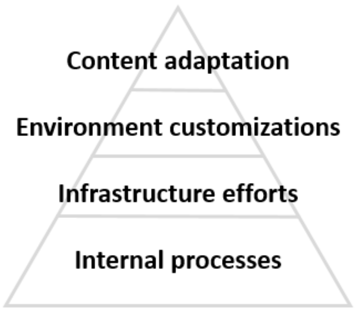
- Stage is the combination of a network, governance rules, data and users in which an environment operates. Typical stages are “Production”, “Development” and “Testing”. An environment operates within one stage. One stage is home to multiple environments.
- Hotfixing is the process of applying SAS® hotfixes. Hotfixes address specific issues but they do not introduce new functionality.
- A release upgrade is the upgrade to a new SAS® release, for example 9.3 to 9.4. Release upgrades introduce major changes to the SAS® landscape of the organization.
- A maintenance level upgrade is the upgrade to a new maintenance level release of the SAS® Platform. An example would be the upgrade from 9.4M4 to 9.4M5. Maintenance level upgrades are not expected to introduce architectural changes but will bring new functionality and new clients. Maintenance level upgrades will also typically bring lots of improvements that has been previously released as hotfixes. Data accesses are not expected to be affected by maintenance level upgrades.

PREPARING FOR THE UPGRADE

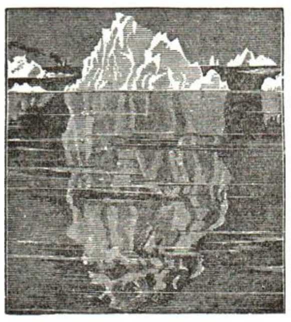
EFFORT ESTIMATION

Depending on your organization, the efforts may look like this or like this:





or maybe even like this:



Effort drivers to consider are:

- People and necessary skills.
- Size and scale of the environments and the resulting complexity (clustering, third party components, etc.)
- Interfaces to third party systems.
- Amount of platform customizations.
- Amount of user generated content. Some content types will require more effort than others. It is therefore crucial to understand your content.
- Multi-tenancy requirements and implementation.
- Time required to test functional usage.
- Company specific processes and regulations.
- Rich clients utilization.
- Infrastructure costs.

UPGRADES ACROSS MULTIPLE STAGES

Most organizations will have multiple stages. A typical multistage setup consists of two to four stages (TEST, PROD or DEV, TEST, PROD or DEV, TEST, PROD, ENGINEERING).

Upgrade strategies may differ across environments and stages. Typically, stages will be upgraded sequentially, and every stage will act as a quality gate for the next one. Effort drivers may be very different across environments: data volumes will be typically higher in PROD, but the amount of user generated content may be higher in DEV because more BI and DI content is generated there. Sometimes access to different stages is limited to specific groups, so there may be one team performing the upgrades in PROD and another for DEV/TEST.

It is helpful if stages are set up according to the same installation blueprint. Often it is tempting to set up the DEV or TEST stages with a lower number of tiers or skip middle tier clustering because it is assumed that this will save costs. But such differences will cause considerably more efforts in the long term because more test cycles will be needed and issues in the upgrade process may surface too late.

How content moves across stages will also have an impact on the upgrade strategy. A conflict may arise in situations where users generate content on an upgraded environment, but must promote it to a not-yet-upgraded environment for testing or production. Different client versions in different stages can lead to corrupted content. The shorter the time window needed to upgrade all stages is, fewer problems will arise from diverging content.

PREPARATION TASKS

a) Asses the current state

To avoid surprises the current state of every environment must be assessed. The assessment should be executed early in the planning stages and verified again before the upgrade. This activity will include tasks like:

- Gather utilization data
- Gather information regarding customizations of the environment
- Check the state of all server contexts on the environment (Stored Process Server, Pooled Workspace Server), as some of them may not be working as expected

b) Understand your users

Depending on the way end users work with the platform, some components should be tested in greater detail. Some users may prefer working on the Unix command line, others may use Enterprise Guide and keep their projects on a department network share. Some do heavy computations and others do simple reporting. Some print their results, others email them or copy and paste them in a spreadsheet.

c) Understand the type of your content

Here is a - by no means complete - list of content types that can be present in your environment. An overview like this is very helpful to clarify the detailed steps for migration and testing.

Content type	**Characteristics**
SAS® Enterprise Guide projects	Can be stored in the filesystem or within the SAS® Content Server.
SAS® Enterprise Miner projects	Cannot be migrated with SPK packages.
DI Content (Libraries, Tables, Jobs, Flows)	May produce exceptionally large SPK packages. In Development environments there may be lots of abandoned content.
BI Content (Information Maps, Web Report Studio reports)	May depend on DI content.
Portal Pages	Cannot be migrated with SPK packages.
Base SAS® Code	SAS® code may contain references to the old environment, like paths, hostnames or metadata-Ids.
Static web content	Can reside on the web server or on the web application server.
Non-SAS content	May be everything from shell scripts to batch job definitions in third party scheduling software.

Table 1. Example content types listing

d) Understand the state of your content

Often there will be lots of outdated or even broken content left over. Users may not know that the content is dysfunctional and will complain about it only after the upgrade.

e) Prepare the depot and hotfix catalogue

A “hotfix freeze” will be issued at a certain point in order to have time for proper testing. No new hotfixes are allowed to be added the upgrade after this point. Hotfixes may have dependencies across tiers.

f) Define and prepare validation procedures

The migrated environment needs to be validated. Technical validation ("shake down tests") are a first step. Much more important and challenging is the validation of the business content. Achieving full test coverage is mostly impossible and too expensive, so clever test procedures must be defined. Here the support of power users ("friends and family") is very valuable. The list of all the aspects that must be tested should be compiled together with the business users, otherwise important features may be missing. Having the validation procedures written down and documented well also allows broader participation in testing.

g) Prepare the SAS® clients rollout

A maintenance release often brings new rich clients. Clients may or may not be compatible across releases. In general, a new set of rich clients must be rolled out. In this paper we will widely neglect the client part of the upgrade, but it is a large topic and it must be considered timely.

MASS UPGRADING

Organizations that operate multiple SAS® platforms (in multiple stages for separate legal or physical entities) should consider establishing upgrade as a bulk process. The automatization of the upgrade process allows not only to decrease the downtime needed, but also to minimize effort and execution errors. An automation tool we have successfully used with SAS is Ansible [1].

REUSABLE TOOLING

A set of reusable artefacts should be created to enable mass upgrades. Such artefacts will be:

- A knowledge base to store many bits and pieces of information
- Detailed step-by-step instructions
- Automation scripts:
    - For creating and preparing environments (think “Infrastructure as Code”).
    - For pre- and post-validation. Automated validation allows us to perform more test cycles in the same time period and will improve the final results.
    - “change package”: templates and code that will drive the upgrade process.
    - For SAS® installation and configuration.
    - For content migration (export/transport/import).

The artefacts should be kept in a version control repository and collaboration tools (e.g. git and Confluence) where all team members have direct access.

As example, consider collecting SAS® Deployment Registry information with Ansible:

```
- name: generate SAS Deployment Registry Report
  shell: /opt/sas/SASPrivateJavaRuntimeEnvironment/9.4/jre/bin/java -jar
sas.tools.viewregistry.jar"
  args:
    chdir: "/opt/sas/deploymntreg"
- name: fetch deployment registry file
  fetch: src="/opt/sas/deploymntreg/DeploymentRegistry.txt" flat=yes
dest="/tmp/fetched/DeploymentRegistry-{{hostname}}.txt"
```

UPGRADE TESTING

Testing upgrades is a destructive process. In order to perform the same test multiple times a mechanism to restore the initial state of the test environment is necessary. How this is best accomplished depends on the realities in your organization. It may involve the use of filesystem snapshots, virtualization snapshots and dedicated clients.

The list of possible tests can include the following:

- Run SAS® OQ tool to verify SAS® Foundation components

  (typically resides in /opt/sas/SASFoundation/9.4/sastest/sasoq.sh)

- Test 3rd party components connectivity (Oracle, DB2, Hadoop, etc.) with prepared libname statements.

- Login to /SASPortal using all configured ways (Kerberos, certificate, form based).

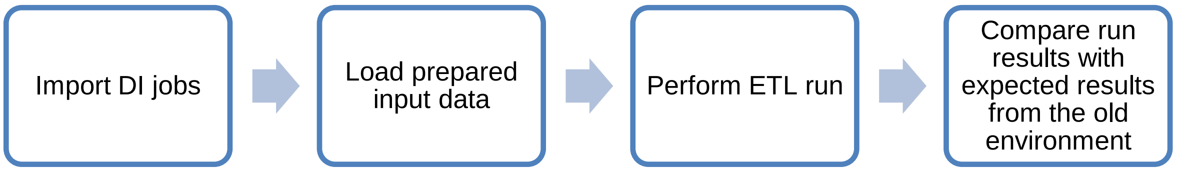
- Test SAS® Web Report Studio by executing a predefined report.

[1] Ansible homepage. Accessed January 2018:
http://ansible.com

- Import and export SPK packages from SMC with end-user rights.
- Test Integrated Windows Authentication with Enterprise Guide.
- If SAS® Enterprise Miner® is used, open a project and execute a predefined diagram. This will perform a full-stack test (Webserver, Web Application Server, compute servers).

Validation of server contexts, including Object Spawner, Stored Process Server, Pooled Workspace Server and Workspace Server can be automated using the SAS® procedure PROC IOMOPERATE.

An example regression test of an ETL flow on an upgraded environment could contain the following steps:


Import DI jobs
Load prepared input data
Perform ETL run
Compare run results with expected results from the old environment

IDENTIFY UNEXPECTED CHANGES

Mass upgrades work smoothly when the source environments are in an expected state. Changes tend to "creep in" over time and even with high level of automation of the change processes it is not guaranteed that all environments will be as expected. Environment differences must be identified during the assessment phase and a plan how to cope with them prepared. Where possible, the environments should be brought to the same base level.

Differences from the expected state that can affect a successful upgrade may be also outside of the SAS Platform. Operating system version, file system sizes and mount points, versions of third party packages should be recorded and verified.

Also, the upgrade scripts must be written in a resilient way. In the Ansible world, the concept of “idempotency” is widely spread. Idempotency is described as follows: *“An operation is idempotent if the result of performing it once is exactly the same as the result of performing it repeatedly without any intervening actions.”* [2] The upgrade scripts shall therefore behave in the same way regardless if they are run multiple times or aborted and restarted.

DEPOT MANAGEMENT

SAS® 9 installation depots are large. Multiple orders and subdepots may have to be considered. We assume that some practices for depot management are already in place in your organization. For a mass upgrade, we must ensure that the proper installation sources are accessible for every stage and environment. This also extends to hotfixes.

In some cases, it may be beneficial to copy the depot locally or at least to a distribution point as near to the environment as possible.

[2] Ansible documentation. Accessed January 2018: http://docs.ansible.com/ansible/latest/glossary.html

PERFORMING THE UPGRADE

UPGRADE IN PLACE

The standard strategy for maintenance level upgrades is “upgrade in place” (UIP). The upgrade is applied directly on the target environment and the user generated content stays where it is. As the infrastructure remains unchanged, there is no need for new servers, DNS (Domain Name System) or firewall changes and content migration.

The upgrade will be performed as follows:

a) Prepare the "change" package

Define all steps necessary to perform the upgrade.

b) Disconnect the users

c) Execute the backup of the platform for each tier

d) Deploy the "change" package

Make sure that we can perform the upgrade. Test free space, expected software levels and configuration. The change package is parametrized, and the target environment is prepared.

Some workarounds may be necessary for the upgrade to run smoothly. Please consider examples below:

- “Temp” folder in the Lev1 folder of the application server. The folder should be moved to backup before starting the upgrade.
- The contents of the “htdocs” folder of the webserver should be moved to backup.
- The start script of the webserver can be exchanged with a “dummy” script to allow the restart of the server during web application server upgrade.

Customizations like special permissions structures, 3rd party components encryption, two-way SSL, TLS for ActiveMQ and similar may have to be reverted to reduce the risk of failure of the upgrade process.

e) Execute the upgrade

The upgrade is executed tier by tier. The proper upgrade in place is performed using SAS® Deployment Manager (SDM). The SDM will take care of the internal processes. Manual execution of SDM allows to fix and retry if issues arise. With increasing quality of the change process such issues should not occur often. Execution time will vary significantly depending on the tier and content. It is recommended to execute backups after successful upgrade of tier. The backups are check- and restart points for the overall process.

f) Post upgrade steps

All customizations must be reapplied and workarounds from the prepare step reverted.

g) Validate the platform and the content

h) Connect the users and enable new rich clients

ALTERNATIVE UPGRADE STRATEGY

An alternative upgrade strategy is “installation & migration” (I&M). A new environment is set up and the user generated content is migrated to it. The decision which strategy is the “right” one depends on several factors:

- Business requirements
- Effort
- Environment constraints
  - Contractual constraints
  - Licensing and legal aspects
- Technical factors
  - Sizing in terms of users, data and content volumes.
  - Complexity of the environment.
  - Support for the deployment strategy. It is possible that some content does not support a specific upgrade strategy very well.
  - Changes in hardware or operating system.
  - Expected life time of underlying infrastructure.

The following table gives an example of a decision matrix on upgrade strategy:

Topic	**I&M Approach**	**UIP Approach**
Infrastructure (Hardware)	Additional resources necessary (at least temporarily)	Can be done without additional resources
Service disruption	Depends *	Depends *
Fallback scenario	Stay on the old environment	Recover from backup
Availability of the old installation after upgrade	Possible	Not possible
Metadata migration path	Export, Import	Not necessary / in place
External data sources and data delivery	Changes to data input systems may be necessary	No change necessary
External data consumers	Changes to data consumers may be necessary	No change necessary
Technical debt	Opportunity to get rid of technical debt that has accumulated in the past	Will likely generate more technical debt
Surface new functionality	Yes	Depends **
Platform Installation	New installation needed ***	Not necessary
Local data	Local data must be moved	Not necessary
Connectivity	Changes to firewall rules and DNS will be necessary	No changes necessary

Table 2. Example decision matrix

*) In a blue/green[3] approach I&M could be faster, but if the environment contains lots of content the time needed to export and import it may be significant.

[3] BlueGreenDeployment, Accessed March 2018: https://martinfowler.com/bliki/BlueGreenDeployment.html

**) In some cases, functionality introduced with the maintenance release may only be available in a new installation. For example, SAS® 9.4 M4 introduced changes to SSL/TLS configuration of the web tier which will not be automatically activated in an in-place upgrade.

***) Installation and configuration of a new environment can be automated to reduce duration and increase quality. Our experience is that automation scripts can be widely reused with small amounts of adaptation across maintenance releases.

INSTALL & MIGRATE

A blue/green approach for the upgrade is being followed. A new environment is set up, the content is migrated to it, tests are executed and only then the users are switched to the new one. The upgrade contains the following main steps:

a) Install target environment with a new maintenance version

Review the paper "Platform à la carte: An assembly line to create SAS® Enterprise BI Server instances with Ansible" [4] for more details about automated installation of the SAS® Platform.

b) Export content from the source environment

SAS® offers the SAS® Migration Utility (SMU) to facilitate the migration of content. SMU supports migration of many types of content, including datasets. Besides SMU, the Platform Object Framework utilities offer mechanisms to perform partial promotion of content using the SPK format.

c) Import and adapt content

Some content may need adaptation on the target environment.

d) Validate content

e) Switch to the new environment

Depending on the duration of the import and validation phases, a final delta deployment of content may be necessary: content which has been created since step b) and cannot be recreated for various reasons. If DNS aliases are used, the final switch will be performed by changing the external DNS aliases to the new servers. Also, the new version of rich clients must be activated accordingly.

HOW CAN I&M BE IMPROVED?

The I&M approach is comparably simple and elegant. Migrating the content fast and without errors may become a challenge. Especially on large environments with many thousands of objects the acceptable time windows may not be sufficient.

It would be very helpful if the user generated content would become easier "pluggable" into new environments by a simple and repeatable process, maybe in an enhanced SMU implementation. This would allow greater flexibility and many scenarios that are not simple to achieve today would become commonplace:

- Move content between on premise and off premise installations to meet peak in demand or try out new software releases.

[4] SAS® Global Forum 2017 Proceedings, Accessed November 2017: https://support.sas.com/resources/papers/proceedings17/0814-2017.pdf

- Establish a new way of platform operation: “rebuild instead of repair”. Highly time and effort intensive repair actions (think web application server debugging, SSL handshake problems and similar) can be exchanged with a new clean installation followed by content migration. Especially cloud-based installations where servers are provisioned in minutes in contrast to weeks on premise can benefit from this approach.
- Long time archiving of regulatory relevant content.

CONCLUSION

Establishing maintenance level upgrades as a cyclic process and part of the service offers the opportunity to plan in a strategic way and to build reusable infrastructure and tooling. Increased quality and shorter time-to-market are an important goal not only for large organizations with dozens of environments, but for also for every provider of SAS® Platforms as a Service.

TAKEWAYS

Content matters

The user generated content is the most valuable asset of your organization. Understanding your content and considering it when planning the upgrade will make the difference between success and failure.

Continuity of change

Change is a constant. In successful organizations changes and upgrades are not something that comes as a surprise but is a part of the offered service.

Architect for change

Organizations that adapt quicker to change will be more successful. Flexibility is a key architecture principle and it should be considered from the beginning.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Javor Evstatiev
javor.evstatiev@evsc.it

Andrey Turlov
andrey.turlov@googlemail.com