snapLogic

Collaborative Development Best Practices

Sharing knowledge to ensure repeatable project success

Executive Summary

This guide is designed for organizations wishing to take full advantage of the potential value of the SnapLogic integration platform through adoption of field-tested best practices for collaborative development. These policies and advice will help users develop smoothly from the technical success of an initial project, to repeatable success of follow-on projects, enabling true digital transformation.

The SnapLogic platform offers a number of capabilities that support the goal of achieving greater organizational maturity and increased collaboration. This document aims to highlight some of these and place them in the correct context for users to be able to determine the precise steps that are most relevant in their own environment and organization.

By implementing these best practices, SnapLogic users in both IT and line-of-business teams will be able to optimize their efforts and ensure the delivery of maximum value from the original investment in SnapLogic.

This document is not an exhaustive how-to guide; the SnapLogic online product documentation remains the authoritative source for specific configurations. In addition, SnapLogic offers a number of consultative programs for customers who are looking to accelerate their digital transformation. Please consult your account manager to find out more about which of these resources may be right for your situation.

Author: **Dominic Wellington**, Enterprise Architect, SnapLogic

Table of Contents

snapLogic

Introduction

The graphical low-code/no-code nature of the SnapLogic development environment makes it easy to get started and build useful automation that solves a user's pressing business problem, lets them answer an urgent question about their data, or enables them to integrate previously disconnected systems via API. The introduction of generative integration through SnapGTP further lowers the barrier to entry, capturing users' intent in natural language and generating the corresponding pipeline or query automatically.

This rapid prototyping and development is key to enabling users to achieve their goals quickly, which in turn ensures rapid return on investment (RoI) for the adoption of SnapLogic technology.

SnapLogic's integration platform as a service (iPaaS) is a full-fledged development environment, but it can sometimes be under-estimated, precisely because of how easy it is to use. IT teams may not think to put in place the sorts of best practices that they would for other, more traditional development tools. In turn, users outside of IT may not have the specific experience to expect such best practices to be implemented.

The risk is that multiple projects may start in parallel and operate in isolation, resulting in duplication of effort and un-optimized use of infrastructure.

SnapLogic architects — employees, partners, and customers — have developed best practices that should be followed to ensure the success of collaborative development projects. These general principles are distilled from hundreds of engagements where users were able to transition from tactical, piecemeal automation to true enterprise integration, enjoying the benefits of that deep architecture work. These recommendations are ready to be adopted and adapted to specific environments and existing policies and standards.

Shared Center of Excellence

The first important recommendation is that a central point of contact be defined for all activities relating to SnapLogic. This group should not be seen as the "owners" of the platform in the sense that they might be with more traditional development environments, developing components in response to requests from users in line-of-business teams. Instead, they should act as expert consultants, combining a close understanding of the wider business priorities with a specific understanding of the local SnapLogic deployment.

The Center of Excellence (CoE) should administer the SnapLogic platform, deploying, scaling, securing, and retiring components as necessary. Beyond these inward-facing tasks, the CoE should also ensure the integration of the SnapLogic platform into existing operational management systems, making logs and events from the SnapLogic infrastructure and pipelines visible and accessible to central SRE teams. Another useful point of integration is version control and backup, ensuring that business-critical integrations developed inside

SnapLogic are managed with the same level of care and attention as those built with more traditional development tools.

While much of the routine maintenance work can be carried out entirely within an IT group, certain actions may require consultation with users. Examples of technical actions to support end-user needs include:

- Scaling the SnapLogic infrastructure to match planned business growth
- Deploying Snaplexes in a distributed fashion to satisfy business requirements (regional access, performance, regulatory or policy compliance)
- Managing credentials and access required for integration with external systems, ensuring those integrations are available to users while also enforcing security mandates
- Reviewing new or proposed developments against best practices
- Developing shared components to avoid duplication of effort and facilitate standardization

The CoE should be staffed primarily by employees of the customer company or long-term consultants who have a deep understanding of the business context within which the SnapLogic platform has been adopted and is expected to operate. Strong relationships with users in line-of-business (LoB) teams will facilitate an ongoing consultative engagement between the CoE and end-users.

Customers who desire specific support may choose to augment their own CoE with a Resident Architect (RA), an expert SnapLogic employee who is assigned to support a specific customer organization. The RA will share best practices and guide the CoE personnel in the most effective deployment and use of SnapLogic technology, ensuring accelerated return on the investment in SnapLogic technology.

Shared Component Library — Design for Reuse

In traditional, text-oriented development, it has been best practice almost since the beginning of programming to establish shared libraries of functionality. With increasing levels of abstraction in the design of programming languages themselves, these shared libraries moved from straightforward re-use of code through copying, to becoming external resources that can be called upon as easily as internal components of a program.

In the same way, mature organizations will ensure that pipelines are created with the expectation of reuse built into their design. These reusable pipelines should be made available as part of a shared component library.

SnapLogic offers a number of Pattern Pipelines for this purpose as part of the Cloud Pattern Catalog. Meanwhile, Pattern Pipelines created within the customer organization will be listed under Projects. In both cases, the Pattern Pipelines are available for immediate use, but will generally require users to supply credentials and potentially some minor environment customization.

This organization-level shared library should be owned by the CoE in order to ensure that expected standards are followed and all required documentation is provided. The CoE may choose to develop components directly, harvest components that have been created for specific purposes and develop them into reusable components, or some combination of the two.

SnapLogic allows for the same mechanisms to be implemented as in any other development suite. The main principles that apply are the following:

- **Modularity**: break complex workflows up into smaller pipelines, reusing as appropriate
- **Loose coupling**: avoid dependencies between pipelines, or on specific environmental concerns
- **High cohesion**: grouping related functionality in a pipeline helps ensure robustness, reliability, reusability, and understandability
- **Information hiding** (encapsulation): design pipelines so that areas expected to change regularly are grouped together to avoid regular wide-ranging changes
- **Separation of concerns**: concentrate interactions with particular third-party systems, in order to facilitate any future changes which might be required (e.g. breaking upgrades or migrations)

Pipeline Nesting

The simplest form of component reuse is nesting pipelines, that is, invoking one pipeline from another. This action relies on the Pipeline Execute Snap, which allows for one pipeline to call another, ensuring that complex or commonly-required functionality can be built once and made available wherever it is needed.

The called pipeline can also execute on a different Snaplex than the parent pipeline, which enables further architectural flexibility. For instance, a pipeline running on less-secure infrastructure such as a DMZ could access functionality that it would not otherwise have direct connectivity with, by invoking a pipeline that executes on secured infrastructure. The benefit here is twofold: the sensitive functionality is only accessed by a single pipeline that can be exhaustively secured and debugged, and network segmentation can be used to limit access to that single point.

Configurable Pipelines

Users can also create an expression library file that contains the details of the environment and import that configuration into pipelines as required. For example, individual teams within the wider organization might have their own specific configuration library with the appropriate settings for the environment. Pipelines can then be moved around from one environment to another without needing to be updated, further facilitating sharing.

The same mechanism should be used to facilitate promotion between development, test, and production environments, without needing to reconfigure the pipelines themselves. These two goals can support one another, further stimulating adoption of best practices.

Error Handling

A particular type of pipeline that may well be overlooked by users without specific development experience is the Error Pipeline. The baseline requirement of this type of pipeline is to handle errors that may occur during execution of the business logic in the main pipeline. However, in a larger and more articulated organization, especially where there is extensive code reuse, error pipelines will also need to be standardized.

Standardization of error pipelines satisfies two requirements:

- **Handling of common error conditions** – It is expected that many pipelines in an organization will interact with common systems and therefore will encounter the same or similar error conditions. These situations should be handled in a uniform fashion, both to avoid duplicate work to handle the same error, and to ensure that the error condition is indeed handled correctly every time it occurs, regardless of which pipeline has triggered it.
- **Easier troubleshooting** – when users are presented with an error message, rapid resolution requires that it be easy to understand and that it be obvious which actions can be taken (and by whom) to resolve the issue. One of the best ways to deliver on this goal of reducing mean time to resolution (MTTR) is to ensure that the same situation will always trigger the same clear and comprehensible error message.

An Error Pipeline is simply a pipeline that is created specifically for handling errors; it processes error documents produced by other Pipelines in the environment. The Error Pipeline runs even if errors are not encountered in the Snaps from the main Pipeline.

Error Pipelines also capture the data payload that triggered the error, which can be vitally important to error resolution. Since the SnapLogic platform will not store customer data, the error pipeline is the most standardized way to capture the actual data which was being processed by the pipeline when the error occurred.

Architecture Best Practices for Collaboration

Environment Separation

It is recommended to create separate environments, at a minimum for development and production, and ideally with an explicit testing area as well. This practice will help to avoid production outages, since errors can be found and corrected in the pre-production environments, before the pipeline is promoted to production where errors would have greater impact.

Different permissions and access controls should be set on the different environments. The development environments should be as open as possible, to enable the maximum number of users to take advantage of the ease of use of the SnapLogic platform to deliver business value. Promotion to testing and production environments on the other hand should be tightly restricted and controlled. SnapLogic offers a powerful hierarchical model of assets and containers, with individual permissions to enable the desired security level to be set for each item.

The Configurable Pipelines feature helps to facilitate promotion of pipelines between environments without the need for manual reconfiguration, which itself would risk introducing errors. By configuring the pipelines in this way, the promotion process can be made entirely automatic. This automation in turn makes it possible to include automated testing and rollback as a routine part of the promotion process.

Operational Best Practices

Versioning and Backup

As the SnapLogic platform is used and new automations are developed, significant business value will be captured within the pipelines that are created. The impact of the loss of any of these developments would be correspondingly large, whether due to human error or technical failure.

The built-in versioning capabilities allows users to replace an existing pipeline with a newer one or to rollback to a previous version in the event that something went wrong. Each version is tagged automatically with information about its creator, as well as the time and date of creation, but it is recommended that users also document the state and purpose of the pipeline in the notes.

It is recommended that users take advantage of these features to safeguard the valuable automations and integrations that users develop over time, and to avoid the negative consequences for the organization that would occur if those were to be lost. The CoE should work to make sure that users without a background in development be made aware of the importance of versioning their work correctly.

In addition to these built-in capabilities, the SnapLogic iPaas offers out-of-the-box integration with industry-standard Git version-control, and specifically the following implementations[1]:

- GitHub
- GitHub Enterprise Server (GHES)
- GitLab.com
- GitLab (self-managed)
- Azure Repos

Through Git integration, users can track, update, and manage versions of SnapLogic project files, pipelines, tasks, and accounts, using either the SnapLogic graphical interface or the SnapLogic Public APIs. These operations include checkout, add, remove, pull, and commit, as well as creating branches and tags.

Git integration supports both Gitflow and trunk-based development; however, SnapLogic recommends trunk-based development.

Simultaneous Edits

It is important to point out that SnapLogic does not currently support simultaneous collaboration in the same pipeline. If one user is editing a pipeline and a different user makes a change to the same pipeline, the original user will be prompted that a new version of the pipeline exists. This is another reason to adopt explicit version management strategies, in order to avoid confusing users who may be used to simultaneous collaborative editing in other tools such as Google Docs.

Secrets Management

Secrets Management is an established best practice in traditional development environments, enabling organizations to use a third-party secrets manager to store endpoint credentials. Instead of entering credentials directly in SnapLogic accounts and relying on SnapLogic to encrypt them, the accounts contain only

[1] List of supported version-control systems accurate as of Fall 2023. Please refer to the SnapLogic online documentation for the current list.

the information necessary to retrieve the secrets. During validation and execution, pipelines obtain the credentials directly from the secrets manager. With Secrets Management, the configured secrets are never stored by the SnapLogic control plane or by Snaplex nodes, facilitating security hardening and simplifying compliance with internal or external policies.

Currently, SnapLogic support integration with the following secrets managers[2]:

- AWS Secrets Manager
- Azure Key Vault
- CyberArk Conjur (Enterprise or Open Source)
- HashiCorp Vault (Cloud, Enterprise, or Open Source)

Observability

SnapLogic offers built-in self-monitoring functionality, including the following capabilities:

- Analyze execution pipelines, including AutoSync data pipelines
- View Snaplex and node metrics

However, generally accepted best-practice for development platforms such as SnapLogic is to integrate them with dedicated monitoring and observability tools. This integration brings the following benefits:

- Avoid relying on the tool to monitor itself
- Reduce Total Cost of Ownership (TCO) by managing monitoring, alerts, and notification system in one place
- Manage the historical retention of data for audit needs
- Correlate data from different tools in order to understand correctly the cause and impact of an issue, avoiding duplication of troubleshooting efforts.

[2] List of supported secrets-management systems accurate as of Fall 2023. Please refer to the SnapLogic online documentation for the current list.

The SnapLogic platform uses OpenTelemetry[3] to support telemetry data integration with third-party Observability tools.

Conclusion

The SnapLogic iPaaS is a full-fledged development environment, and should be considered as such when it comes to both management of assets developed within the platform, and industry-standard integration with third-party tools.

The ease of use of SnapLogic naturally lends itself to wide adoption by users outside of IT, and this trend is only expected to accelerate further with the introduction of generative integration via SnapGPT. The speed with which these users can achieve their goals and deliver business value may however be overlooked by centralized IT teams, leading them to underestimate the potential business value that can be achieved through implementation of industry-standard best practices for development environments.

Fortunately, most of these well-understood best practices — collaborative development, modular architectures, staged infrastructure, and integration with operational management tools — are available out of the box with SnapLogic. All that is required is active engagement between core IT, the CoE team responsible for the SnapLogic, and the end-users themselves, who are after all the ones closest to the business value that is being delivered by the platform.

[3] As of Fall 2023, this feature is available as a beta release, enabling monitoring of Pipeline execution runtime logs in Datadog.