GenAI Best Practices for SnapLogic - A comprehensive reference guide

Best Practices for Adopting AI Solutions in the Enterprise with SnapLogic Agent Builder

Introduction: AI in the Modern Enterprise

AI is fast becoming a cornerstone of modern enterprises, transforming how businesses operate, make decisions, and interact with customers. Its capabilities, such as automation, predictive analytics, and natural language processing, allow companies to streamline processes, gain deeper insights from data, and enhance customer experiences. From optimizing supply chains to personalizing marketing strategies, AI is enabling enterprises to innovate, drive efficiency, and be competitive in an increasingly data-driven world. As AI continues to evolve, its role in shaping business strategy and operations will only grow.

Precisely because of its novelty and importance, leaders will need to think carefully about various aspects of how these powerful new capabilities can be deployed in a manner that is compliant with existing legislation and regulation, and how best to integrate them with existing systems and processes. There are no generally-accepted best practices in this field yet due to its novelty, but there are lessons that we can learn from past waves of technological change and adoption. In this document we set out some suggestions for how to think about these topics in order to ensure a positive outcome.

Data

Data is the lifeblood of IT — arguably the reason for the field's entire existence — but its importance is only magnified when it comes to AI. Securing access to data is a requirement for an AI project to get off the ground in the first place, but managing that access over time, especially as both the data and the policies that apply to it change and evolve over time.

Data Security and Management

Data security has always been a complex issue for IT organizations. When considered from the perspective of AI adoption, two main areas need to be considered: external and internal usage. Externally-hosted Large Language Models (LLMs) offer powerful and rapidly-evolving capabilities, but also have inherent risks as they are operated by

third parties. The second area of focus is how and what internal data should be used with AI models, whether self-managed or externally operated.

External Security

Organizations have reasonable concerns about their proprietary, regulated, or otherwise sensitive information “leaking” beyond the organizational boundaries. For this reason, simply sending internal information to a public LLM without any controls in place is not considered a viable solution.

An approach to this problem that was previously considered promising was to use a technique called Retrieval Augmented Generation, or RAG. In this approach, rather than passing user queries directly to an LLM for answers, a specialized data store is deployed, called a vector database. When a user query is received, the vector data store is consulted first to identify relevant *chunks* of information with which to answer the query, and only after this step is the LLM used to provide the conversational response back to the user.

However, while RAG does limit the potential for information leakage, it does not reduce it to zero. The vector database can be operated according to the organization’s own risk profile: fully in-house, as a private cloud instance, or leveraging a shared platform, depending on the information it contains and the policies or regulations that apply to that information. However, a chunk of information will be sent from the vector store to the LLM to answer each query, and over time, this process can be expected to expose a substantial part of the knowledge base to the LLM.

It is also important to be aware that the chunking process itself still uses a LLM. More security-sensitive organizations or those operating in regulated industries may choose to leverage a more restricted deployment model for the LLM as well, much as discussed for the vector database itself, in order to avoid this leakage. However, it is worth noting that while an “open-source” language model can be prevented from contributing training data back to its developers, its own pre-existing training data may still leak out into the answers. The ultimate risk here is of “model poisoning” from open-source models. That is, injection of data from outside the user’s domain which may lead to inconsistent or undesirable responses. One example of this phenomenon is “context collapse”, which may occur in the case of overloaded acronyms, where the same acronym can represent vastly different concepts in different domains. A generalist model may mis-understand or mis-represent the acronym — or worse, may do so inconsistently.

The only way to be entirely certain of data security and hygiene is to train the model from scratch — an undertaking that, due to its cost in both time and resources, is

practical only for the largest organisations, and is anyway required only for the most sensitive data sets.

A halfway house that is suitable for organisations that have concerns in this domain, but not to the point of being willing to engineer everything themselves from the ground up, is *fine-tuning*. In this approach, a pre-trained model is further trained on a specific data set. This is a form of *transfer learning* where a pre-trained model trained on a large dataset is adapted to work for a specific task. The dataset required for this sort of fine-tuning is very small compared to the dataset required for full model training, bringing this approach within reach of far more organizations.

Internal Data Access Controls

The data that is consumed by the AI model also needs to be secured inside the organization, ensuring that access controls on that data follow the data through the system at all levels. It is all too easy to focus on ingesting the data and forget about the *metadata*, such as role-based access controls. Instead, these controls should be maintained throughout the AI-enabled system. Any role-based access controls (RBAC) that are placed on the input data should also be reflected in the output data.

Agentic approaches are useful here, as they give the opportunity to enforce such controls at various points. The baseline should be that, if a user ought not be able to access certain information through traditional means such as database queries or direct filesystem access, they also must not be able to access it by querying an AI overlay over those systems — and vice-versa, of course.

Prompt Logging and Observability

An emerging area of concern is the security of prompts used with AI models. Especially when using public or unmodified open-source models, the primary input to the models is the prompt that is passed to them. Even minor changes to that prompt can cause major differences in what is returned by the model. For this reason, baseline best practice is to ensure that prompts are backed up and versioned, just as would be done for more traditional program code. In addition, both prompts and their corresponding responses should be logged in order to be able to identify and troubleshoot issues such as performance changes or impact to pricing models of public LLMs. Some more detailed suggestions are available here.

Prompts should also be secured against unauthorized modification, or “prompt injection”. Similarly to the analogous “SQL injection”, attackers may attempt to modify or replace the

prompt before it is passed to the AI model, in order to produce outputs that are different from those expected and desired by users and operators of the system. The potential for damage increases further in the case of agentic systems that may chain multiple model prompts together, and potentially even take actions in response to those prompts. Again, logging for both in-the-moment observability and later audit is important here, including the actual final prompt that was sent to the model, especially when that has been assembled across multiple steps. These logs are useful for troubleshooting, but may also be formally required for demonstrating compliance with regulation or legislation.

Example Prompt Injection Scenarios

Direct Injection

An attacker injects a prompt into a customer support chatbot, instructing it to ignore previous guidelines, query private data stores, and send emails, leading to unauthorized access and privilege escalation.

Indirect Injection

A user employs an LLM to summarize a webpage containing hidden instructions that cause the LLM to insert an image linking to a URL, leading to exfiltration of the private conversation.

Unintentional Injection

A company includes an instruction in a job description to identify AI-generated applications. An applicant, unaware of this instruction, uses an LLM to optimize their resume, inadvertently triggering the AI detection.

Intentional Model Influence

An attacker modifies a document in a repository used by a Retrieval-Augmented Generation (RAG) application. When a user's query returns the modified content, the malicious instructions alter the LLM's output, generating misleading results.

Code Injection

An attacker exploits a vulnerability in an LLM-powered email assistant to inject malicious commands, allowing access to sensitive information and manipulation of email content.

Payload Splitting

An attacker uploads a resume with split malicious prompts. When an LLM is used to evaluate the candidate, the combined prompts manipulate the model's response, resulting in a positive recommendation regardless of the resume's actual contents.

Multimodal Injection

An attacker embeds a malicious prompt within an image that accompanies benign text. When a multimodal AI processes the image and text concurrently, the hidden prompt alters the model's behavior, potentially leading to unauthorized actions or disclosure of sensitive information.

Adversarial Suffix

An attacker appends a seemingly meaningless string of characters to a prompt, which influences the LLM's output in a malicious way, bypassing safety measures.

Multilingual/Obfuscated Attack

An attacker uses multiple languages or encodes malicious instructions (e.g., using Base64 or emojis) to evade filters and manipulate the LLM's behavior.

reference: https://genai.owasp.org/llmrisk/llm01-prompt-injection/

As these examples show, there are many patterns and return sets from LLMs that will need to be managed and observed, comparing prompts, responses, and data sets with certified sets and expected structures. Hopefully over time and as commercial LLMs mature, many of these issues will be managed by the LLMs themselves, but today these concerns will have to be part of the enterprise's own governance framework for AI adoption.

Data Ownership and Observability

Much of the value of most Generative AI (GenAI) applications is based on the quantity, freshness and reliability of the source data that is provided. An otherwise fully-functional GenAI tool that provides responses based on incomplete or out-of-date data will not be useful or valuable to its users.

The first question is simply how to gain access to useful source data, and to maintain that access in the future. This work spans both technical and policy aspects. SnapLogic of

course makes technical connectivity easy, but there may still be questions of ownership and compliance, not to mention identifying where necessary data even resides.

Beyond the initial setup of the AI-enabled system, it will be important to maintain ongoing access to up-to-date data. For instance, if a RAG approach is used, the vector data store will need to be refreshed periodically from the transactional data platform. The frequency of such updates will vary between use cases, depending on the nature of the data and its natural rate of change. For instance, a list of frequently asked questions, or FAQs, can be updated whenever a new entry is added to the list. Meanwhile, a data set that is updated in real time, such as airline operations, will need much more frequent synchronization if it is to remain useful.

Recommendations

Data is key to the success of AI-enabled systems – and not just one-time access to a dataset that is a point-in-time snapshot, but ongoing access to real-time data. Fortunately, these are not new concerns, and existing tools and techniques can be applied readily to securing and managing that flow of data. In fact, the prominence and urgency of AI projects can even facilitate the broad deployment of such tools and techniques, where they had previously been relegated to specialized domains of data and analytics.

It is important to note that as the SnapLogic platform facilitates connectivity and movement of data, none of the patterns of such movement are used to enrich the learnings of a LLM. In other words, pipelines act to transport encrypted data from source to destination without any discernment of the actual payload. No payload data and no business logic governing the movement of data are ever gleaned from such movement or used to train any models. In fact, the SnapLogic platform can be used to enhance data security at source and destination as highlighted above, adding guardrails to an AI system to enforce policies against publication of sensitive or otherwise restricted data.

In general it is recommended for domain experts, technical practitioners, and other stakeholders to work together and analyze each proposed use case for AI, avoiding both reflexive refusals and blind enthusiasm, focusing instead on business benefit and how to achieve that in a specific regulatory or policy context.

Auditability and Data Lineage

The ability to audit the output of AI models is a critical requirement, whether for routine debugging, or in response to incoming regulation (e.g. the EU AI Act) that may require auditability for the deployment of AI technology in certain sectors or for particular use cases. For instance, use of AI models for decision support in legal cases or regulated industries, especially concerning health and welfare, may be subject to legal challenges, leading to requests to audit particular responses that were generated by the model. Commercial and legal concerns may also apply when it comes to use cases that may impinge on IP protection law.

Complete forensic auditability of the sort that is provided by traditional software is not possible for LLMs, due to their non-deterministic nature. For this reason, deterministic systems may still be preferable in certain highly-regulated spaces, purely to satisfy this demand. However, a weaker definition of auditability is becoming accepted when it comes to LLMs, where both inputs and outputs are preserved, and the model is required to provide the source information used to generate that output. The source data is considered important both to evaluate the factual correctness of the answer, and also to identify any bias which may make its way into the model from its source data.

These factors make auditability and data lineage critical part of the overall AI Strategy, which will have to be applied at various different stages of the solution lifecycle;

- **Model creation and training** - This aspect relates to how the model was created, and whether the data sets used to train the model have a risk of either skewing over time, or exposing proprietary information used during model development.
- **Model selection** - The precise version of AI model that was used to generate a response will need to be tracked, as even different versions of the same model may produce different responses to the same prompt. For this reason it is important to document the moment of any change in order to be able to track and debug any drift in response or behaviour. For external third-party AI services, these models may need to be tested and profiled as part of both an initial selection process and ongoing validation The reality is that there are is no single AI that is best for all use cases. Experience from real-world deployment of actual AI projects shows that some models are noticeably better for some functions than others, as well as having sometimes radically different cost profiles. Thiese factors means that most probably several different AI models will be used across an enterprise, and even (as agents) to satisfy a single use case.

- **Prompt engineering** - Unlike in traditional software development, by its very nature, prompt engineering includes the data sets and structures in the development cycle in a way that traditional functional coding practices do not. The models' responses are less predictable, so understanding how the data will be processed is an integral part of the prompt engineering lifecycle. To understand how and why a set of prompts are put into production will be driven by both the desired functionality and the data that will be provided, in order to be able to review these inputs if issues arise in the production environment.
- **Prompt evaluation in production** - All critical systems today should have robust logging and auditing processes. However many enterprises rarely achieve universal deployment, and coverage is often inconsistent. Due to the nature of AI systems, and notably LLMs, there will be a critical need to audit the data inputs and outputs. The model is effectively a black box that is not available for operators to reconstruct exactly why a given response was provided. This issue is especially critical when multiple AI models, agents, and systems are chained together or networked within a wider process. Logging the precise inputs that are sent to the model will be key for all these purposes.

For custom-trained models, these requirements may also extend to the training data — although this case is presumed to remain relatively rare for the foreseeable future, given the prohibitive costs of performing such training. Where more common approaches (RAG, fine-training) are used that do not require an entire model to be trained from scratch, the audit would naturally focus on the inputs to the model and how those are managed. In both these cases, good information hygiene should be maintained, including preservation of historical data for point-in-time auditability.

Backing up the data inputs is necessary but not sufficient: after all, a different LLM (or a subsequent version of the same LLM) may provide different responses based on the same prompt and data set. Therefore, if a self-trained LLM is employed, that model should also be backed up in the same way as the data that feeds it. If a public LLM is used, rigorous documentation should be maintained identifying any changes or version upgrades to the external model. All of this work is in addition to the tracking of the prompts and data inputs themselves, as described previously.

All of these backups will in turn need to be preserved according to whatever evidentiary concerns are expected to apply. In the case of simple technical audits to ensure continuous improvements and avoid downward pressure on the quality of responses provided, organizations can make their own determination on the level of detail, the width of the time window to be preserved, and the granularity of the data. In more highly regulated scenarios, some or all of these elements may be mandated by outside

parties. In those situations, the recommendation would also generally be to specify the backup policy defensively, to avoid any negative impacts in the case of future challenges.

Development and Architecture Best Practices

While AI systems have notable differences from earlier systems, they are still founded in large part on pre-existing components and techniques, and many existing best practices will still apply, if suitably modified and updated.

CI/CD

Continuous Integration and Continuous Deployment is of course not specific to Generative AI. However, as GenAI projects move from demo to production, and then evolve over subsequent releases, it becomes necessary to consider them as part of that process.

Many components of a GenAI application are stateful, and the relationship between them can also be complex. A roll-back of a vector data store used to support a RAG application may have unforeseen effects if the LLM powering that RAG application remains at a different point of the configuration timeline.

Therefore the different components of an AI-enabled system should be considered as tightly coupled for development purposes, as otherwise the GenAI component risks never becoming a fully-fledged part of the wider application environment. In particular, all of the traditional CI/CD concepts should apply also to the GenAI component:

- Continuous Development
- Continuous Testing
- Continuous Integration
- Continuous Deployment
- Continuous Monitoring

Ensuring the inclusion of development teams in the process is unlikely to be a problem, as the field of AI is still evolving at breakneck pace. However, some of the later stages of an application's lifecycle are often not part of the worldview of the developers of early demo AI applications, and so may be overlooked in the initial phases of productization of GenAI functionality. All of these phases also have specific aspects that should be

considered when it comes to their application to GenAI, so they cannot simply be integrated into existing processes, systems, or modes of thought.

New aspects of development and DevOps are needed to support, notable prompt engineering will have to be treated as code artifacts, but will also have to be associated with prompts such as model version, test data sets and samples of return data so that consistent functional management of the combined set of capabilities can be understood and tracked over time.

QA and Testing

Quality Assurance (QA) and testing strategies for AI projects, particularly GenAI, must address challenges that differ significantly from traditional IT projects. Unlike traditional systems where output is deterministic and follows predefined rules, GenAI systems are probabilistic and rely on complex models trained on vast datasets. A robust QA strategy for GenAI must incorporate dynamic testing of outputs for quality, coherence, and appropriateness across a variety of scenarios. This involves employing both automated testing frameworks and human evaluators to assess the AI's ability to understand prompts and generate contextually accurate responses, while also mitigating risks such as bias, misinformation, or harmful outputs.

A GenAI testing framework should include unique approaches like model evaluation using synthetic and real-world data, stress testing for edge cases, and adversarial testing to uncover vulnerabilities such as the attack scenarios listed above. Frameworks such as CI/CD are essential but need to be adapted to accommodate iterative model training and retraining processes. Tools like Explainable AI (XAI) help provide transparency into model decisions, aiding in debugging and improving user trust. Additionally, feedback loops from production environments become vital in fine-tuning the model, enabling ongoing improvement based on real-world performance metrics rather than static, pre-defined test cases. However, depending on the use case and the data provided, such fine-tuning based on user behavior may itself be sensitive and need to be managed with care.

The QA process for GenAI also emphasizes ethical considerations and regulatory compliance more prominently than traditional IT projects. Testing needs to go beyond technical correctness to assess social impact, ensuring that the system avoids perpetuating harmful bias or misinformation.

Continuous monitoring after deployment is crucial, as model performance can degrade over time due to shifting data distributions. This contrasts with traditional IT projects,

where testing is often a finite phase before deployment. In GenAI, QA is an evolving, lifecycle-long endeavor requiring multidisciplinary collaboration among data scientists, ethicists, domain experts, and software engineers to address the complex, dynamic nature of generative models.

Grounding, as an example, is a technique that can be used to help produce model responses that are more trustworthy, helpful, and factual. Grounding generative AI model responses means connecting them to verifiable sources of information. To implement grounding, usually means retrieving relevant source data. The recommended best practice is to use the retrieval-augmented generation (RAG) technique. Other test concepts include.

- Human-in-the-Loop Testing: Involves human evaluators judging the quality, relevance, and appropriateness of the model's outputs. Source data accuracy with details and data.
- Adversarial Testing: Actively trying to "break" the model by feeding it carefully crafted inputs designed to expose weaknesses and vulnerabilities.

Deployment

This aspect might superficially be considered among the easiest to cover, but that may well not be the case. Most CI/CD pipelines are heavily automated; can the GenAI aspects be integrated easily into that flow? Some of the processes involved have long durations, e.g. chunking a new batch of information; can they be executed as part of a deployment, or do they need to be pre-staged so that the result can simply be copied into the production environment during a wider deployment action?

Monitoring

Ongoing monitoring of the performance of the system will also need to be considered. For some metrics, such as query performance or resource utilization, it is simply a question of ensuring that coverage also spans to the new GenAI experience. Other new metrics may also be required that are specific to GenAI, such as users' satisfaction with the results they receive. Any sudden change in that metric, especially if correlated with a previous deployment of a change to the GenAI components, is grounds for investigation. While extensive best practices exist for the identification of technical metrics to monitor, these new metrics are still very much emergent, and each organization should consider carefully what information is required — or is likely to be required in the event of a future investigation or incident response scenario.

Integration of AI with other systems and applications

AI strategies are already moving beyond pure analytic or chatbot use case, as the agentic trend continues to develop. These services, whether home grown or hosted by third parties, will need to interface with other IT systems, most notably business processes, and this integration will need to be well considered to be successful. Today LLMs is producing return sets in seconds, and though the models are getting quicker, there is a trend to trade time for greater resilience of quality. How this trade-off is integrated into high performance business systems that operate many orders of magnitude faster will need to be considered and managed with care. Finally, as stated throughout this paper, AI's nondeterministic nature will mandate a focus on compensating patterns across the blend of AI and process systems.

Recommendation

While it is true that the specifics of AI-enabled systems differ from previous application architectures, general themes should still be carried over, whether by analogy, applying the spirit of the techniques to a new domain, or by ensuring that the more traditional infrastructural components of the AI application are managed with the same rigor as they would be in other contexts.

Service / Tool Catalog

The shift from content and chatbot experiences to agentic approaches imply a new fundamental architectural consideration of which functions and services should be accessible for use by the model. In the public domain there are simple patterns and models will mainly be operating with other public services and sources — but in the enterprise context, the environment will be more complex. Some examples of questions that a mature enterprise will need to address to maximize the potential of an agentic capability ;

> **What are the "right" services?** Large enterprises have hundreds, if not thousands, of services today, all of which are (or should be) managed according to what their business context is.
>
> A **service management catalog** will be key to manage many of these issues, as this will give a consistent point of entry to the service plane. Here again, pre-existing API

management capabilities can ensure that the right access and control policies can be applied to support the adoption of composable AI-enabled applications and agents.

When it comes to **security profiling** of a consuming LLM, the requester of the LLM service will have a certain level of access based on a combination of user role and security policy that is enforced. The model will have to pass on this to core systems at run time so that there are no internal data breaches.

When it comes to **agentic systems**, new questions arise, beyond the simpler ones that apply to generative or conversational applications. For instance, should an agent be able to change a record? How much change should be allowed and how will this be tracked?

Regulatory Compliance

While the field of GenAI-enabled applications is still extremely new, best practices are beginning to emerge, such as those provided by the Open Web Application Security Project (OWASP). These cybersecurity recommendations are of course not guaranteed to cover any particular emerging regulation, but should be considered a good baseline which is almost certain to give a solid foundation from which to work to achieve compliance with national or sector-specific regulation and legislation as it is formalized.

In general, it is recommended to ensure that any existing controls on systems and data sets, including RBAC and audit logs, are extended to new GenAI systems as well. Any changes to the new components — model version upgrades, changes to prompts, updates to training data sets, and more — will need to be documented and tracked with the same rigor as established approaches would mandate for traditional infrastructure changes.

The points made previously about observability and auditability all contribute to achieving that foundational level of best-practice compliance. It is worth reiterating here that full coverage is expected to be an important difference between GenAI and previous domains. Compliance is likely to go far beyond the technical systems and their configurations, which were previously sufficient, and to require tracking of final prompts as supplied to models, including user input and runtime data.

Conclusion

Planning and managing the deployment and adoption of novel AI-enabled applications will require new policies and expertise to be developed. New regulation is already being created in various jurisdictions to apply to this new domain, and more is sure to be added in coming months and years. However, much as AI systems require access to existing data and integration with existing systems to deliver value at scale, existing policies, experience, and best practices can be leveraged to ensure success. For this reason, it is important to treat AI as an integral part of strategy, and not its own isolated domain, or worse, delegated to individual groups or departments without central IT oversight or support.

By engaging proactively with users' needs and business cases, IT leaders will have a much better chance of achieving measurable success and true competitive advantage with these new technologies — and avoiding the potential downsides: legal consequences of non-compliance, embarrassing public failures of the system, or simply incorrect responses being generated and acted upon by employees or customers.