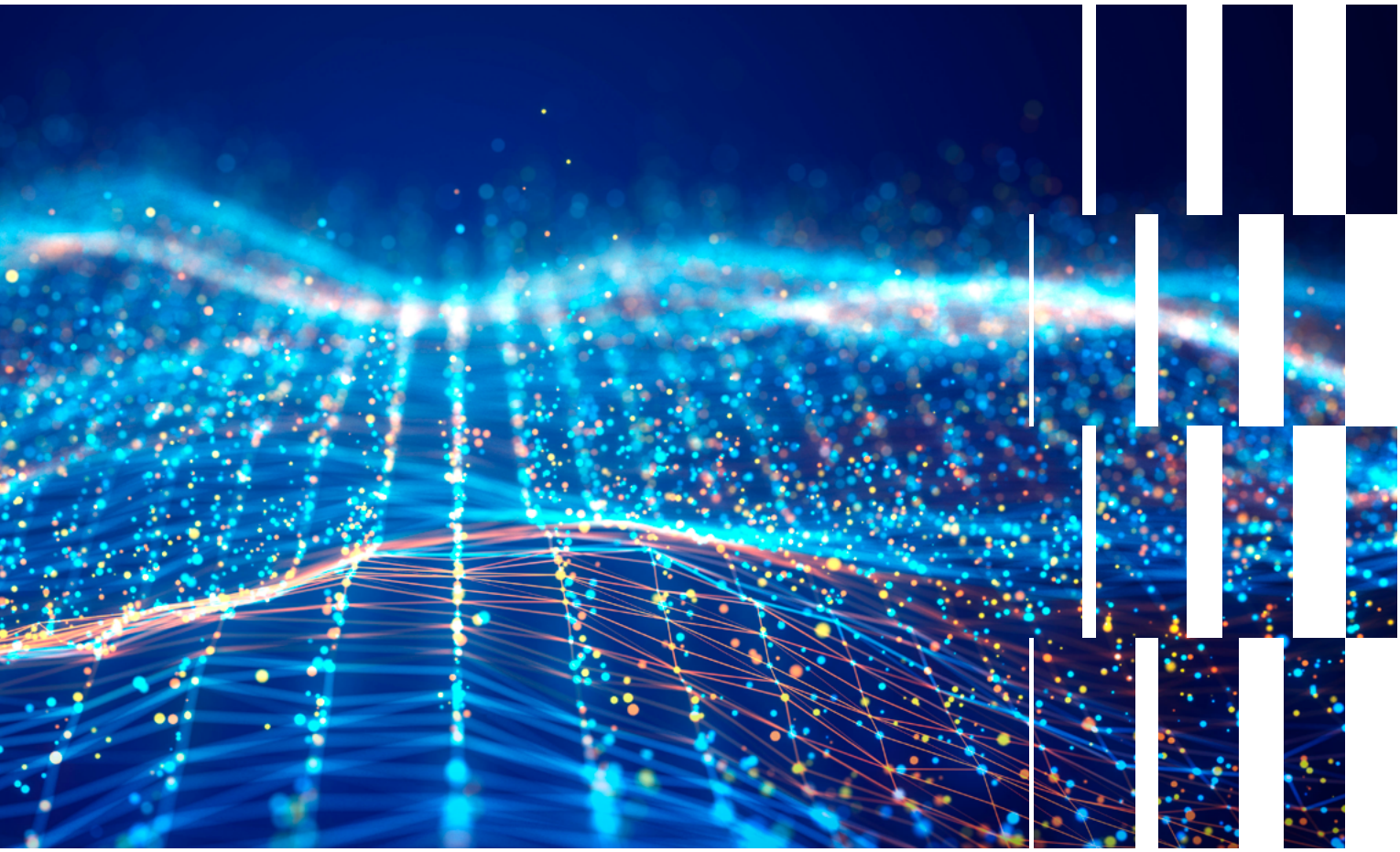
DELL Technologies



WHITEPAPER

High Velocity IT Service Delivery with IaC, GitOps and ITSM

This white paper describes successful approaches to blending infrastructure as code (IaC) and GitOps in traditional environments, such as ITSM. Outcomes include accelerating infrastructure changes, empowering innovation, speeding delivery of high-value services and products, and increasing operational efficiency. These approaches also meet high standards for governance, security, and reliability.

TABLE OF CONTENTS

Executive summary

In today's increasingly digital economy, velocity and innovation are top requirements for business success. In response, more enterprises are adopting modern automation approaches, such as infrastructure as code (IaC) and GitOps.

The challenge has been blending these modern approaches with traditional ones, such as information technology service management (ITSM). At Dell Technologies Services, we have advised many of our customers on how to maintain ITSM's governance, security, and high availability while dramatically improving speed of infrastructure change and operational efficiency.

IaC and GitOps focus on describing the desired infrastructure state and automating changes in a secure, standardized fashion. This is a shift from ITSM-based step-by-step processes to implement changes. In this white paper, we present several solutions that combine modern DevOps with traditional approaches and deliver high-impact business outcomes.

Introduction

For decades, many large organizations have used ITSM to provide consistent, secure, and reliable procedures for delivery of IT services. More recently, enterprises have sought a leaner, automated approach to scaling and accelerating infrastructure change.

Some enterprises that have supplemented ITSM with IaC tools and GitOps processes have run into roadblocks. While IaC automates and completes change processes in minutes, ITSM governance procedures still can add weeks or months to fulfilling change requests. When this occurs, friction can develop between ITSM and automation teams.

At Dell Technologies Services, we've discovered through engagements with numerous customers that it doesn't have to be this way. In fact, an IT strategy that marries the best of ITSM and IaC that delivers optimal outcomes of faster service delivery, improved scalability, and better ROI can be a reality, while still meeting top-tier data security and application availability requirements.

IaC translates configurations that describe the desired state of the infrastructure in a declarative fashion. GitOps defines the governance and automated processes that enable the desired state. Together, IaC and GitOps enable standardization and repeatability of testing and rapid pushing of code to production.

IaC and GitOps differ from ITSM, which offers prescriptive step-by-step processes to reach the desired state. Because GitOps often does not offer a robust governance model for larger organizations, we recommend adapting IaC and GitOps to ITSM principles.

Introducing IaC and GitOps requires a shift in thinking from viewing the true state of production operations as Git-based instead of the configuration management database (CMDB). The CMDB, which generally lags the Git production state by a few minutes or less, should continue to run certain

activities. Combining CMDB with Git and IaC will provide a stronger, more agile infrastructure.

When delivering IT services, technology limitations are rarely critical path. For example, it only takes minutes to spin up a virtual machine (VM), yet users may need to wait weeks due to approval processes and backlogs of requests.

With an environment that successfully blends ITSM with IaC and GitOps, organizations gain the ability to scale to hundreds of infrastructure changes per day instead of weeks or months. Automated change processes empower teams to expand their role from implementing changes and servicing tickets to focusing on innovation and software development aligned to business-critical priorities.

These capabilities help enterprises stay ahead of the curve in markets and technologies that evolve faster than ever before. They also maintain high governance standards, minimize security risks, and reduce costs through improved productivity and efficiency.

Legacy approaches to change management

In traditional IT environments, changes are typically triggered by a request to perform an operation on a target infrastructure, such as a create operation or a modify operation to an infrastructure component. Typically, an operations team responds to these requests and performs them manually or by running scripts. Changes would require overlay processes to ensure scalability and compliance with an organization's security and governance requirements.

Many large organizations have embraced ITSM solutions, such as ServiceNow and other tools. ITSM incorporates several elements, including:

- **Service desk serving as a single point of contact between users and IT staff**
- **Change management processes for service requests**
- **Service catalog identifying pre-defined service elements**
- **Well-defined incident management**
- **Provisioning of a CMDB to store configuration items and their relationships**

In an ITSM environment, IT operations typically use transactional semantics to respond to change request-related procedures, such as:

- **Catalog service consumption events that typically create a change request**
- **Triggering of approvers, change logs, and other processes**
- **Pre-defined steps charting the change request**
- **Execution of the change is reflected in the CMDB or as a live service instance**

Accelerating innovation with IaC

One approach to automating infrastructure changes uses the code base to apply changes to the infrastructure and treats the logic as a software artifact. This way, DevOps and continuous integration / continuous delivery (CI/CD) enable quick and rapid changes to the code.

The second approach focuses on executing infrastructure changes, which we refer to as infrastructure state changes or simple state changes. Here, IT uses CI tools to test state changes in a sandbox before the CD process applies them to the target environment.

IaC is growing in popularity as a viable alternative to traditional infrastructure operations. IaC defines the infrastructure desired state with human and machine-readable configuration files in YAML, JSON or domain-specific languages (DSLs). IaC employs automation to alter the desired state of infrastructure. Automation also reconciles the changed infrastructure with the desired state.

Adopting IaC allows changes to be made to infrastructure by stipulating desired state in the form of a configuration file, similar to a contract. The configuration is interpreted by the underlying tools, and abstracts direct API calls, making the process far easier than manually writing logic for operations against APIs. In addition, concepts like idempotency, state tracking and fault tolerance make the overall process of automating infrastructure changes far easier and more manageable.

With IaC, change requests typically focus on altering the desired state configuration files, which speeds adoption of infrastructure changes. This IaC process, described as declarative operations, is distinguished from procedural operations triggered by requests to change the infrastructure in traditional settings.

GitOps, an IaC function and DevOps tool, uses Git repositories to store desired state configuration files, as well as governance and audit processes, as an alternative to the CMDB approach. In GitOps, pull requests control change requests, providing branching models for concurrent changes, approvers overseeing control gates prior to merges, and final status of the most recent change.

In addition, GitOps combines the principles of ITSM and IaC with declarative operations. GitOps is becoming more widely used among developers due to its simplicity and the role of the final status of the most recent change in problem resolution.

Workflows for procedural versus declarative operations

Interactions for infrastructure changes in procedural operations differ from those in declarative operations.

In their respective flows, both approaches interact with the same objects, such as the person requesting the change, an approver who reviews the change, the

operator who executes the change, and the system storing the infrastructure configuration.

In declarative operations, the requester asks for a change to the desired state that can be pulled and applied to the target system. This approach simplifies the overall flow by removing the need to report or fix any discrepancies when applying changes to the desired state. A declarative approach is designed to make systems consistent.

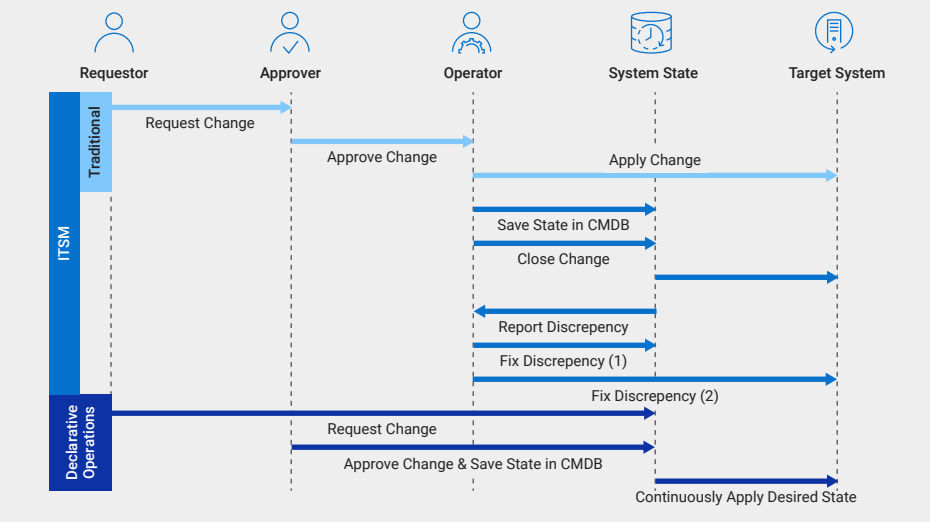


Figure 1: Declarative approach to infrastructure changes

CMDB data quality challenges

In ITSM and ITIL environments, IT may provision a CMDB to address a lack of unified, consistent descriptions of configuration items (CIs). The challenge is that while the CMDB facilitates operations and reports on the state of the system, only the configured infrastructure reflects the true system state. For example, an accurate number of disks in a system requires counting the disks because the CMDB, which is a copy, may contain stale data about the system state.

As a result, many organizations lack confidence in their CMDBs due to stale data. To resolve this issue, IT may implement auto-discovery of elements and relevant discrepancies. This is a complex process requiring incident creation to resolve any inconsistencies.

A better way to tackle data inconsistencies involves viewing a CMDB as storing the desired state of the system rather than the actual system state and accepting temporary inconsistencies. This way, the infrastructure eventually will align to a desired state, either through automation or manual intervention. In addition, CMDB reports, which reflect the desired state, are acceptable since the infrastructure will reach the desired state in a relatively short period of time. The CMDB drives infrastructure change, rather than serving as a by-product of infrastructure operations.

Another challenge occurs when multiple CMDBs are created due to introducing

declarative operations, such as GitOps, in a regulated organization. An example of such a situation may include one traditional CMDB in ServiceNow and an additional CMDB serving as the CI inside the Git repository. Multiple versions of truth across different CMDBs creates problems that the original ITSM CMDB was intended to resolve. An even more complex situation is when multiple CMDBs reflect varying degrees of accuracy due to real-time representation of underlying service instances. Here, a complex correlation of the different databases is the only way to generate a true picture.

At Dell Technologies Services, we recommend operating from a single source of truth based on the desired state rather than the current state. The actual state of any databases should be viewed as a snapshot in time of the current state. Traditional principles of data warehousing, such as the warehouse is valid for the snapshot in time, should apply. To avoid data duplication, an organization that adopts GitOps in full form should treat Git as the source of truth and other databases as relevant only for reporting. We have successfully implemented this approach in large customer environments with relative ease.

Transitioning from procedural to declarative requests

Organizations that adopt modern practices need to evolve from a procedural operational approach of creating, deleting, and updating transactional operations to declarative operations based on change requests that alter the desired state. This shift is not trivial since procedural approaches are often based on years or even decades of implementation.

Additionally, organizations need to trust that automation will ensure that infrastructure systems will eventually reach the desired state. Building this trust requires a governance process to ensure that automation is secure and adheres to all organizational requirements. In addition, the automated development cycle needs to incorporate the same level of testing as with any other software development. Organizations should expect that it will be harder to build assurance relating to custom requests with complex convergence patterns versus standard pre-defined changes.

If the target infrastructure diverges from the desired state by manual intervention or an unforeseen event, automated self-healing will ensure the desired state is reached. By overriding changes that conflict with the desired state, automation also improves overall infrastructure security. If automation is not able to reach the desired state, organizations will be notified, and a service incident should be created.

IaC and Git challenges in ITSM environments

One challenge with IaC is configuring the code to be both machine and human readable. Further, users need enough technical proficiency to be familiar with GitOps.

Only users with technical knowledge should perform infrastructure changes to ensure the environment is properly defined. To help simplify infrastructure change, we also recommend using custom, simple to use tools and user

interfaces to generate machine readable configuration files in the Git repository. To successfully manage these tools and automated processes, we have implemented ServiceNow catalog items as a form builder that acts as a front end to the Git pull request process.

GitOps is a relatively young and increasingly popular discipline. Even as GitOps evolves, several of the tools are limited. To query configuration items and relationships in Git to produce reports, for example, a custom tool would need to be developed due to a non-standardized format of the desired state and the location of the Git tool set. The tool suites for traditional CMDBs, which are typically stored in RDBMS's, are more advanced. To address the limitations, we recommend organizations apply their own standards to Git tools, as well as other as implement monitoring with real-time alerting and operational support tooling.

In an organization with a well-developed ITSM practice, a service desk typically serves as a single point of contact for users. Not only does a service desk streamline access to services but it enables uniformity across services. In GitOps, a service desk does not exist, and users must access multiple repositories. Further, a custom format for a particular Git project requires understanding the related branches, folders, and JSON, YAML, or domain-specific language (DSL) formats.

To address this complexity, we suggest implementing enterprise-wide standards and uniform tools, such as search engines and portals fronting Git repositories. For our customers, we have successfully built a service catalog serving as a simple interface to underlying Git repositories. We also recommend generating architectural blueprints that cover applications' underlying infrastructure, so a single request seeds all services.

In an organization that already has a well-established request approval process, introducing Git pull requests can create a layer of complexity. Additionally, pull requests require Git technical knowledge, such as branches and merge operations. If adopting GitOps, we recommend shifting the approval process for IT operations to the pull request model to help simplify processes.

Alternative solutions for declarative operations

There are three viable solutions to consider when adopting a declarative approach to infrastructure operations. Each of these solutions provides different advantages depending on requirements of the enterprise.

ITSM CMDB

Here, the ITSM CMDB is treated as the single source of truth for the desired state of the infrastructure. An ITSM CMDB is most suited for environments with low change rates, limited requirements for scaling users and services, and high frequency of custom requests. Some of the challenges include slow development cycles and time to market, monolithic implementations, CMDB consistency, and rollback requirements.

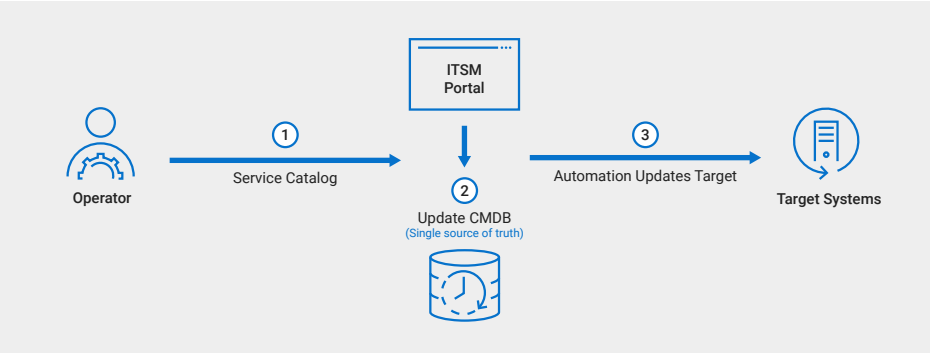


Figure 2: Using an ITSM CMDB as single source of truth

Unlike procedural operations, desired configurations are stored in the CMDB. These configurations are then applied and reconciled with declarative code.

A change request is complete when the desired state is approved and stored within the CMDB. Automation ensures that the infrastructure reaches this desired state. For example, this approach can be implemented with ServiceNow as the ITSM portal and the Ansible module to extract the ServiceNow CMDB data.

GitOps and ITSM CMDB

Git repositories should be treated as an integral part of an ITSM CMDB implementation. GitOps works best for environments with high change frequency, fast development cycles, technically proficient users, site reliability engineering (SRE) teams, rollback requirements, and standardized services.

To successfully integrate GitOps with ITSM CMDB, an organization will need to facilitate a mindset shift, map existing governance requirements to GitOps, manage the impact of ITSM asset management, create a single and centralized service desk, and standardize service offerings. In addition, limited off-the-shelf tools will require more internal development.

Git repositories control versions of the desired state configurations and IaC CD pipelines apply them to the target systems. A pull request for review and approval must handle any modification to the desired state. Pre-flight checks and self-healing help reduce numbers of incidents. In addition, incidents should be exceptions and can be raised using existing ITSM processes.

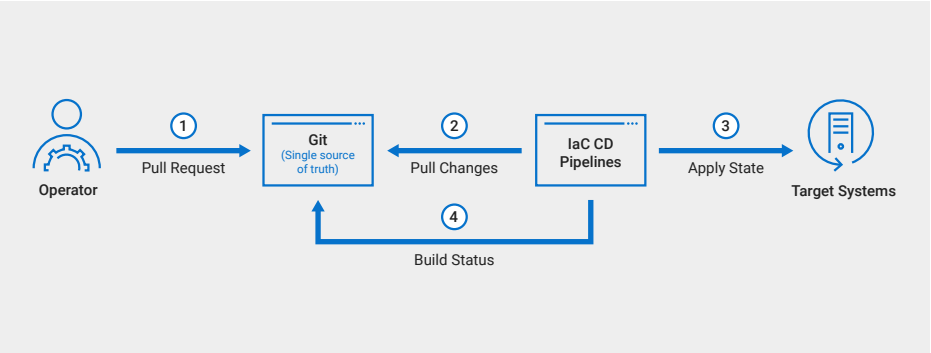


Figure 3: GitOps approach to infrastructure changes

Hybrid: GitOps, IaC and CMDB

A hybrid implementation duplicates configuration data across Git repositories and the CMDB and is designed to make the CMDB and Git configuration data eventually consistent. This deployment is ideal for CMDBs with external dependencies, and environments distinguished by frequent changes, fast development cycles, technical end-users, SRE teams, and rollback requirements.

Like GitOps, organizations will need to manage a mindset shift, integrate existing governance requirements, operate a single service desk, and standardize service offerings.

Git repositories describe the desired state of the target systems and CMDB stores snapshots of the target system state. The ITSM change process is implemented using GitOps pull requests. Any existing reports can be generated from the state databases.

The hybrid solution is suitable for organizations that have heavily invested in tools, such as ServiceNow, for their ITSM services. GitOps can be implemented for selected use cases and provide a periodic snapshot of the current state, which is also shown in the CMDB.

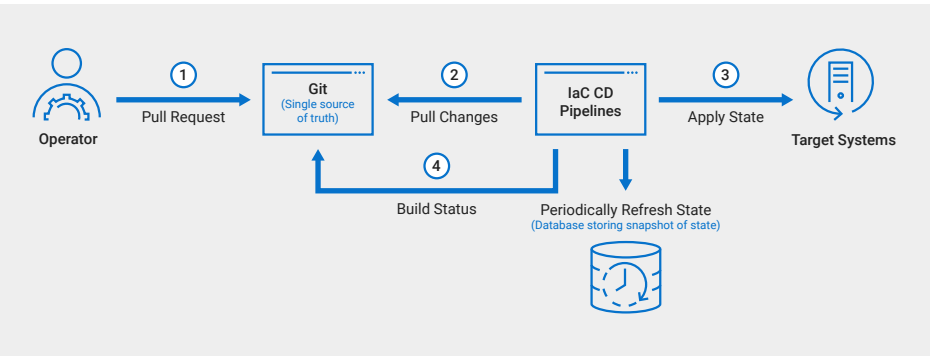


Figure 4: Hybrid GitOps and CMDB approach to infrastructure changes

Future-proof your infrastructure with IaC

In this white paper, we have proposed solutions for enterprises that would like to accelerate their IT operations with IaC and provide increased agility and operational efficiency. The best approach to meet fast evolving requirements of today's enterprises is a system based on infrastructure configuration files that define desired state and automation that applies this desired state to the IT infrastructure.

We also recommend implementing declarative operations for ITSM where the CMDB should be a repository of the desired state rather than a snapshot of the current state. We describe solutions for enterprises with an existing ITSM, as well as organizations seeking to adopt ITSM. The integration of GitOps with the current generation of operations tools is critical to accelerating adoption and implementation of IaC.

Pros and Cons

	Change Management	Incident	CMDB Data	Best For	Challenges
ITSM CMDB as source of truth	Existing ITSM	Existing ITSM	Existing ITSM	Low change frequency Limited scale (e.g. users, services and target systems) High volume of custom requests	Fast development cycles Time to market Monolithic implementation CMDB consistency Rollback
GitOps	GitOps Pull request	Reduce volume of incidents with pre-flight checks and self-healing Incidents are exceptions and raised using existing ITSM processes	Git repositories	High frequency of change Fast development cycle Technical end-users SRE teams Rollback Standard service offering	Mindset shift Mapping existing governance requirements (e.g. security, approval) to GitOps Siingle service desk Limited off-the shelf tooling Impact on ITSM assets management Standardize service offering
Hybrid: GitOps with accompanying state databases	GitOps pull request	Reduce volume of incidents with pre-flight checks and self-healing Incidents are exceptions and raised using existing ITSM processes	Git repositories Duplicating configuration data (Git + CMDB)	External dependencies to CMDB High frequency of change Fast development cycle Technical end-users SRE teams Rollback	CMDB data is eventually consistent Mindset shift Mapping existing governance requirements (security, approval) to GitOps Single service desk Standardize service offering