WHITEPAPER

Cloud **Deployment Patterns**

Exploring the Transformation of the Enterprise Cloud Journey

sourced
cloud at scale™

Table of **Contents**

What is a Cloud Deployment Pattern?

A **Cloud Deployment Pattern** defines an approach to provisioning and managing the lifecycles of resources in the cloud. It is often one of the most critical decisions that underpins and influences an organisation's cloud adoption journey.

Successful adoption of such deployment pattern for an organisation is made in consideration of the following factors:

Security and Compliance Controls

Target Operating Model

End User Experience

Organisational Capabilities

An organisation may adopt a deployment pattern based on the requirements, skills and capabilities at the time and gradually evolve or supplement their approach to cloud consumption as they familiarise themselves with new services, tools, and techniques.

Organisations often adopt multiple deployment patterns, enabling application teams of varying knowledge and skill levels to migrate to public cloud at their own pace and comfort level.

Avoiding the “Free for All Sprawl”

The **“free for all sprawl”** frequently occurs when an organisation enables unrestricted access to services and resources in a cloud environment. Runaway resources are provisioned often through uncontrolled “Click Ops” console interaction.

Without overarching governance or usage guidelines, this commonly results in:

Extensive cost increases in the environment

Ineffective cost reporting and attribution

Lack of visibility and understanding of services and resources in use

Inability to identify owners for provisioned resources

Difficulty in evaluating the security posture of the environment

Costly audit and remediation initiatives to bring the environment into compliance

Although this approach might be suitable for experimentation, training, and evaluation purposes, it **does not** lend itself well to production use cases at scale.

Infrastructure as Code

One of the common better practices for modern technology organisations is to define, run and update infrastructure resources as code.

Taking form of a structured language such as **JSON**, **YAML** or **HCL**, it is used to describe and enforce the desired state of the infrastructure for the target cloud environment. Tools and services including CloudFormation or Terraform are often synonymous with this approach and are often combined with Deployment Pipelines as the foundation of a deployment pattern.

Deployment Pipelines

Deployment Pipelines encapsulate the automated and repeatable steps required to create, update or terminate cloud resources defined as code.

Enabling a range of capabilities, the pipelines are often used as a more scalable alternative for provisioning cloud resources. The complexity of each Deployment Pipeline varies significantly based on the tooling, methodology or the significance of the deployment requirements.

Continuous Integration & Deployment (CI/CD)

Maintaining configuration as code enables use of familiar software development practices, including automated testing, Continuous Integration (CI) and Continuous Deployment (CD) for the infrastructure components of an application.

Understanding **Common Deployment Patterns**

Working with a variety of organisations has allowed us to observe trends in convergence to a set of Cloud Deployment Patterns over a number of years.

To help guide our clients, we have identified some common deployment patterns used in organisations today.

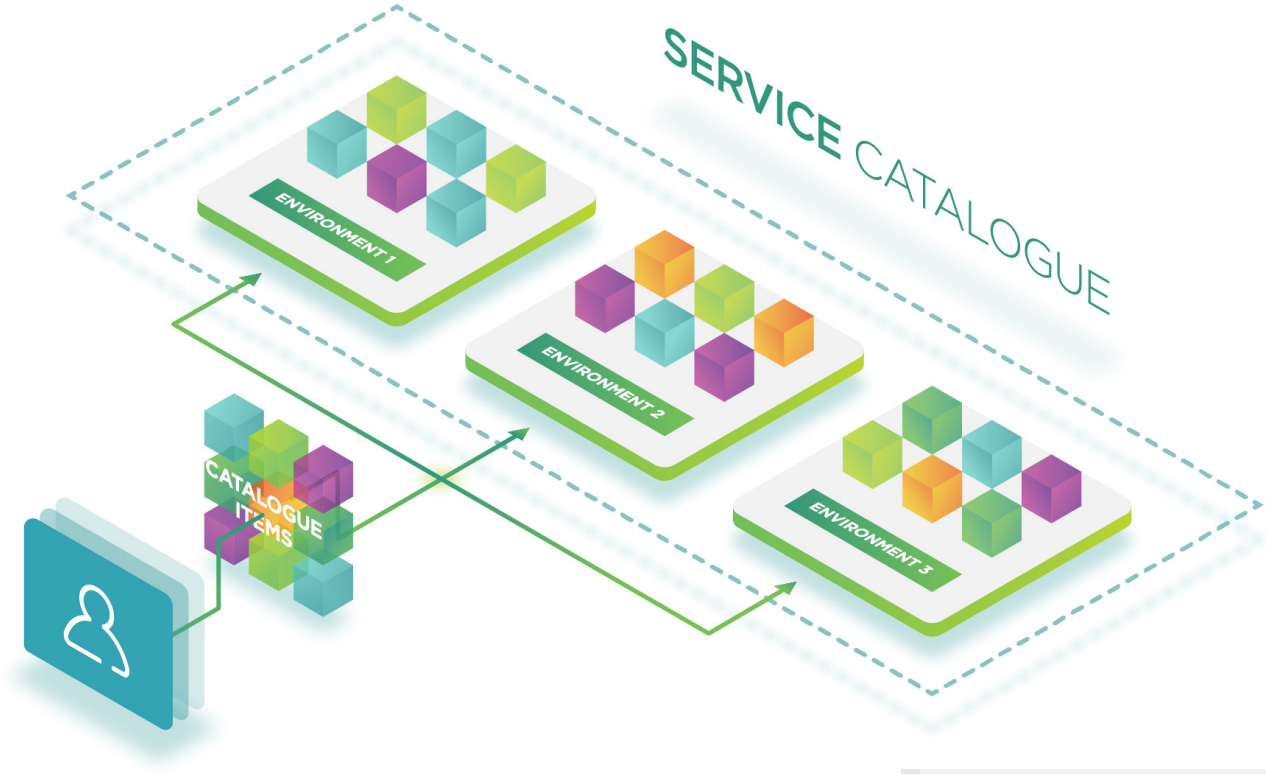
Each pattern comes with potential benefits and downsides and should be carefully evaluated against the key factors of **security controls**, **target operating model**, **end-user experience** and **organisational capabilities**.

Service Catalogue

The **Service Catalogue** pattern is based on consumption of approved cloud patterns as catalogue items which can be "requested" through a user interface, resulting in provisioning and configuration of resources and services in a target cloud environment.

Under this more restrictive pattern, end users cannot access cloud services or tools outside of the catalogue. As a result, this model is often adopted by organisations with more traditional IT processes and capabilities that lend themselves towards consuming **Infrastructure as a Service** (IaaS).

The organisation's cloud platform team has complete control over operational, security and compliance measures. This makes the model suitable for long-running workloads or common deployment patterns that change infrequently.



DOWNLOAD THE DIAGRAM HERE

This deployment pattern exhibits the following characteristics and items for consideration during evaluation:

TARGET OPERATING MODEL

- It reduces the barrier to entry due to the limited direct user interactions to the public cloud platform
- There is a low operational overhead generated for maintaining existing and developing additional catalogue items and services

END USER EXPERIENCE

- The Developer flexibility is limited, as the full range of services the cloud provider offers is inaccessible unless exposed through a developed catalogue item
- This model can be implemented as part of a Deployment Pipeline workflow; however, the catalogue often provides limited automation processes and capabilities to developers
- This method abstracts away the native cloud experience as developers will only leverage the functionality exposed by service catalogue items and their limited APIs

SECURITY AND COMPLIANCE

- Provides high confidence that the provisioned resources are aligned to the target security controls and remain at the defined standards

ORGANISATIONAL CAPABILITIES

- This model can grow increasingly difficult as more complex, non-IaaS services are offered with their own unique lifecycles and workflows

Technical implementations of this deployment pattern include:

- AWS Service Catalog
- GCP Private Catalog
- Azure Managed Applications
- ServiceNow with automated AWS provisioning workflows
- Cloud broker provisioning platforms
- Amazon Managed Services (AMS)
- In-house provisioning interfaces backed by CloudFormation templates or direct API calls

Infrastructure Module Pipeline

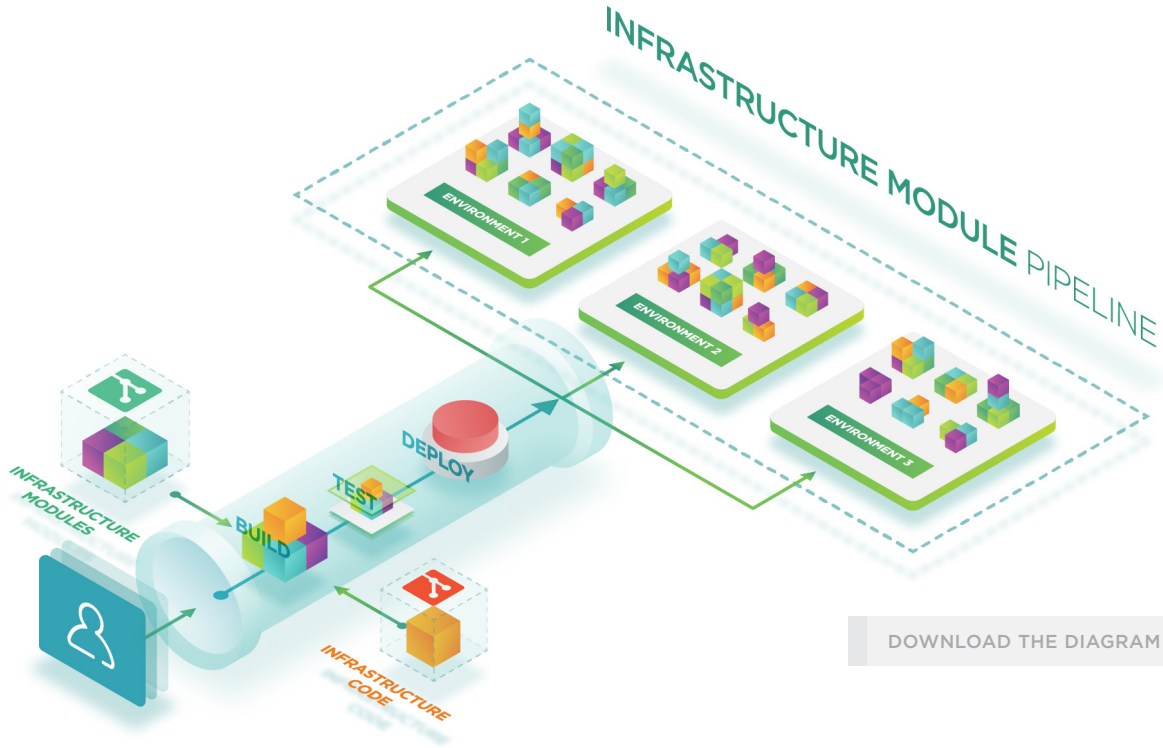
The **Infrastructure Module Pipeline** is an Infrastructure as Code (IaC) deployment pattern based on codifying organisational opinions and architectures of how cloud services should be configured, and providing them to end users as a library of modular patterns. End users provision applications by "assembling" these modules as required, deploying them through a centralised deployment pipeline.

Under this model, end users can only use modules offered in the library, ensuring that all provisioned cloud resources tightly align to the organisation's security and operational standards. The deployment pipeline in this case acts as a gatekeeper to deployments, failing those requests that are not leveraging modules from a common library.

In the context of this model, organisational opinions on cloud resources could include:

- All compute instances must write system logs to a centralised log store
- All databases must be encrypted with a specific encryption key
- All resources must follow a set naming and tagging format
- All Kubernetes pods must contain a security sidecar container

An implementation of this pattern is shown in the diagram below. User defined infrastructure code is combined with a set of authorised modules or patterns maintained by the organisation at the deployment time. After further testing and validation, the combined infrastructure-as-code definitions are submitted to the Cloud Providers API for provisioning, resulting in a deployment that is compliant with the organisation's operational, reliability and security controls.



DOWNLOAD THE DIAGRAM HERE

The Terraform Module Library Pattern

Terraform modules for common resources and services are developed that align to the organisation's standards. These modules are published and distributed to the internal teams for consumption through a registry or a set of common version-controlled repositories.

The end users build solutions using these modules, submitting the deployments through a common CI/CD tool chain.

The logic in the Deployment Pipeline evaluates the code, ensuring that all modules being used are sourced from a library of approved patterns.

Non-compliant deployments are stopped and prevented from occurring.

The Consumable Template Library Pattern

In this implementation, a set of structured templates for common resources and services are provided to the end users as a documented Domain Specific Language (DSL).

The end users build solutions using these templates, submitting deployments through a common CI/CD tool chain where cloud provider provisioning templates representing their solution are generated.

Upon successful template generation, it is then submitted to the cloud provider's API's on behalf of the end user for provisioning.

Although similar to the Terraform Modules, DSL based consumable patterns tend to be a lot more bespoke and abstract than the implementation of the pattern away from the consumers, often requiring just few parameters to achieve a highly sophisticated deployment that might also include lifecycle management of the resources.

This deployment pattern exhibits the following characteristics and items for consideration during evaluation:

TARGET OPERATING MODEL

- Provides a library of approved architectural patterns maintained by a central cloud team or as inner-sourcing practice
- Provides a way to abstract cloud provider level complexities away from end users who may not be experts, letting them focus on application deployment rather than provider intricacies
- Organisations may also be locked into specific deployment Pipeline tooling which becomes coupled with this deployment pattern, resulting in a “platform lock-in”

END USER EXPERIENCE

- The library of consumable modules or templates allows developers to rapidly assemble common patterns across the organisation and deploy applications faster and more consistently
- The standardised modules, templates and pipelines must be developed and implemented prior to the developers / end users being able to build and deploy applications
- Does not always align with services that do not support Infrastructure as Code practices (i.e. Chat Bots, Internet of Things, Augmented Reality, etc)
- Depending on the implementation, there can be a perception from end users that they are building a non-portable skill set as it is linked to an organisation-specific tool-chain or closed source templating language (DSL) that they cannot learn or experiment with outside the organisation

SECURITY AND COMPLIANCE

- Provides high confidence that the provisioned resources are aligned to, and remain at defined standards

ORGANISATIONAL CAPABILITIES

- The standardised deployment workflows can greatly accelerate migration as it is adopted across an organisation, while secure consumable resource deployments leverage approved, well-architected patterns

Inline Validation Pipeline

The **Inline Validation** model is a deployment pattern that leverages a centralised pipeline for consumers within the organisation. At the core of this pattern is an enforced inspection process established for validation of any provisioning requests - only fulfilling them in the event that the request is free of errors and misconfigurations.

This approach is generally implemented through organisational convergence and adherence to a common IaC templating language (such as Resource Manager, CloudFormation, Deployment Manager templates or Hashicorp Terraform), and relying on mandatory pipeline stages for performing static analysis on the request prior to provisioning.

It is the responsibility of the organisation's technology and security teams to ensure that there are suitable checks implemented within the pipeline. It is critical to ensure there is coverage for all services and configuration properties of interest.

Requests that fail due to misconfiguration report responses about issues that must be addressed in the request for it to pass validation and progress to the provisioning phase.

Using this approach, operational, compliance and security opinions can be enforced including:

- Do the resources in the template align to the group's naming convention?
- Are compute instances or containers attempting to be deployed from images that contain known vulnerabilities?
- Are there roles that do not include appropriate constraints to ensure a least privilege model?
- Is there network configuration that permits traffic from excessive address ranges such as 0.0.0.0/0?

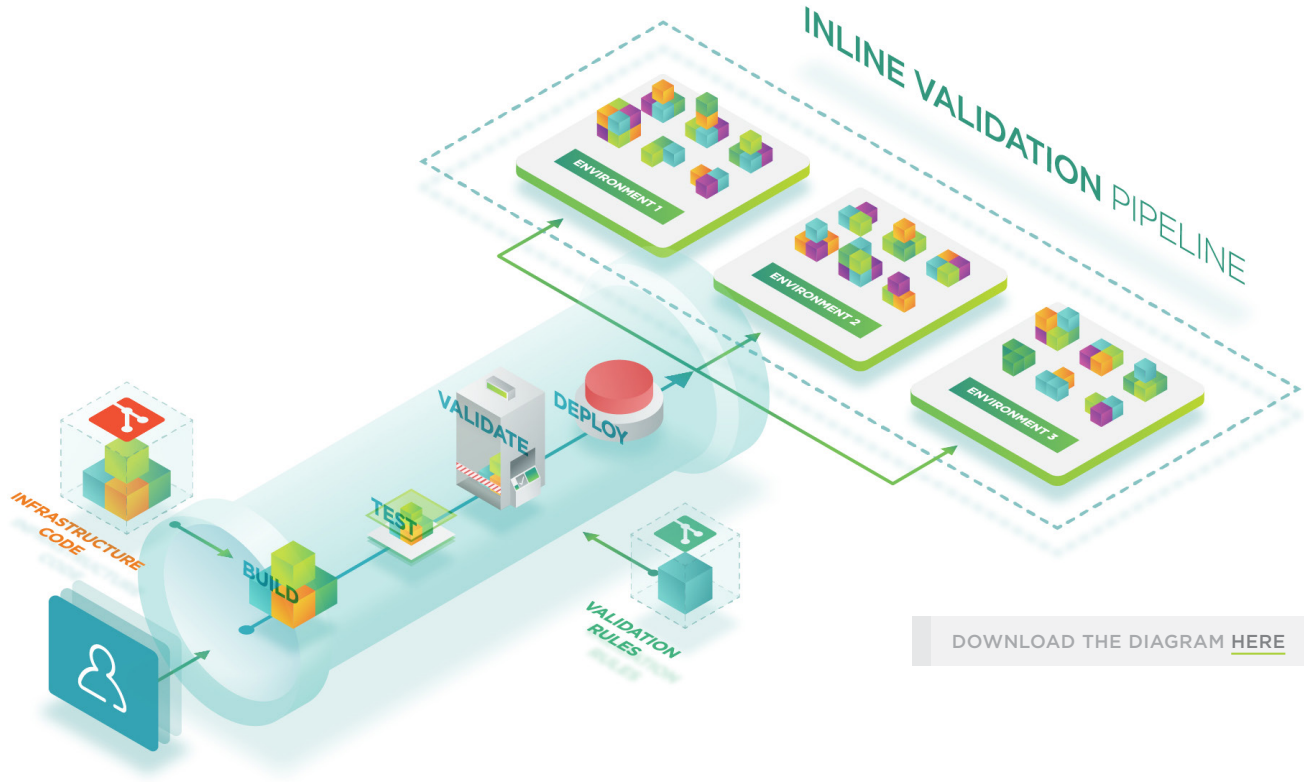
An implementation of an inline validation pipeline is shown in the diagram below. The provisioning templates are submitted to a centralised pipeline that uses a static analysis tool to check that the contained resources align to the required controls.

IF THE VALIDATION STAGE IS PASSED SUCCESSFULLY:

Templates are passed through to the next stage for provisioning.

IF THE VALIDATION STAGE FAILS:

Templates are returned to the end user with error messages containing guidance of changes that are required to align to the required controls and standards.



DOWNLOAD THE DIAGRAM HERE

This deployment pattern exhibits the following characteristics and items for consideration during evaluation:

TARGET OPERATING MODEL

- Operational standards can be enforced with intelligent feedback loop through the inspection of cloud resources properties and patterns

END USER EXPERIENCE

- Implementation through infrastructure-as-code static analysis and linting provides developers access to native tooling and use of direct and native cloud services

SECURITY AND COMPLIANCE

- Security and Compliance standards can be enforced with intelligent nagging through the inspection of cloud resources properties and patterns
- Due to the comprehensive nature of these validation checks, policies must be generic enough to address common configurations but aim to avoid hindering development
- Languages that allow embedding of programmatic code such as custom resources or serverless code can be challenging to inspect with confidence
- Over time, this model risks the loss of granularity as cloud services provisioning requests require further inspection of additionally released configuration attributes requiring the development of new inline rule sets prior to provisioning

ORGANISATIONAL CAPABILITIES

- This model requires organisational convergence on one provisioning language or engine, such as Resource Manager, CloudFormation or Deployment Manager templates or Hashicorp Terraform which may not be suitable for all application types
- This model also requires centralised and mandatory Deployment Pipeline tool to be an effective control

Some example tools that provide inline validation capabilities include:

› Terraform Enterprise with Sentinel

An embedded policy-as-code framework, integrated with HashiCorp Enterprise Products, such as Terraform Enterprise. It enables fine-grained, logic-based policy decisions at run time.

› OPA Policy checks against AWS CDK resources

› cfn_nag

A tool that looks for patterns in CloudFormation templates that may indicate insecure infrastructure, such as IAM rules or security groups that are too permissive, encryption or access logs that aren't enabled.

› Static Analysis of Azure ARM Templates Using Microsoft Pester

Freedom with Guardrails

Unlike the previously described approaches, the **Freedom with Guardrails** model enables end users to access and deploy workloads directly, utilising Cloud Providers' APIs and native deployment capabilities without the requirement of mandated tooling.

Organisational opinions and architectural patterns are not directly enforced during the development life cycle unless they are implemented by the team themselves.

Organisations adopting this approach require their end users to be proficient at foundational cloud knowledge and possess firm understanding of well architected principles and security best practices for cloud adoption.

This ensures that the workloads deployed under this model align to both the provider recommendations and the business' security and compliance guidelines.

High-level preventative controls (**Guardrails**) are applied to the environments, providing broad, coarse-grained controls, limiting some of the actions that may increase security or operational risk for the environment, while still providing high levels of flexibility to the end users for deploying resources and using cloud services.

These controls might provide the following functionality:

- Allowing the use of networking resources within an account whilst preventing the modification of this configuration, thereby retaining authority of network resources by a central team
- Allowing the creation of compute instances whist denying the ability to expose them directly to the internet
- Allowing the use of services in one particular geographic region and denying their use in all others due to data sovereignty concerns
- Allowing the provisioning of object storage resources whilst limiting the ability to make the objects stored within them accessible to the public

In addition to the preventative guardrails, organisational compliance monitoring and reporting is implemented with continuous compliance tools and services. These tools leverage detective capabilities to analyse the state of the resources deployed in the environment, compare them to the organisation's security or compliance baselines and trigger alerts to notify suitable teams when misconfigured resources have been deployed.

Example continuous compliance rules might include:

- Ensuring that there is no network configuration that permits traffic from excessive address ranges such as 0.0.0.0/0
- Ensuring that roles include appropriate constraints to ensure a least privilege model
- Ensuring there are no storage devices, compute, or database instances that are not using encryption
- Ensuring that object storage services are not open to the public
- Ensuring that load balancers do not have incorrect TLS settings

In addition to reporting on the deviation from the required baseline, some governance tools have the ability to remediate the issue through the deletion of the resource or applying a suitable control to it.

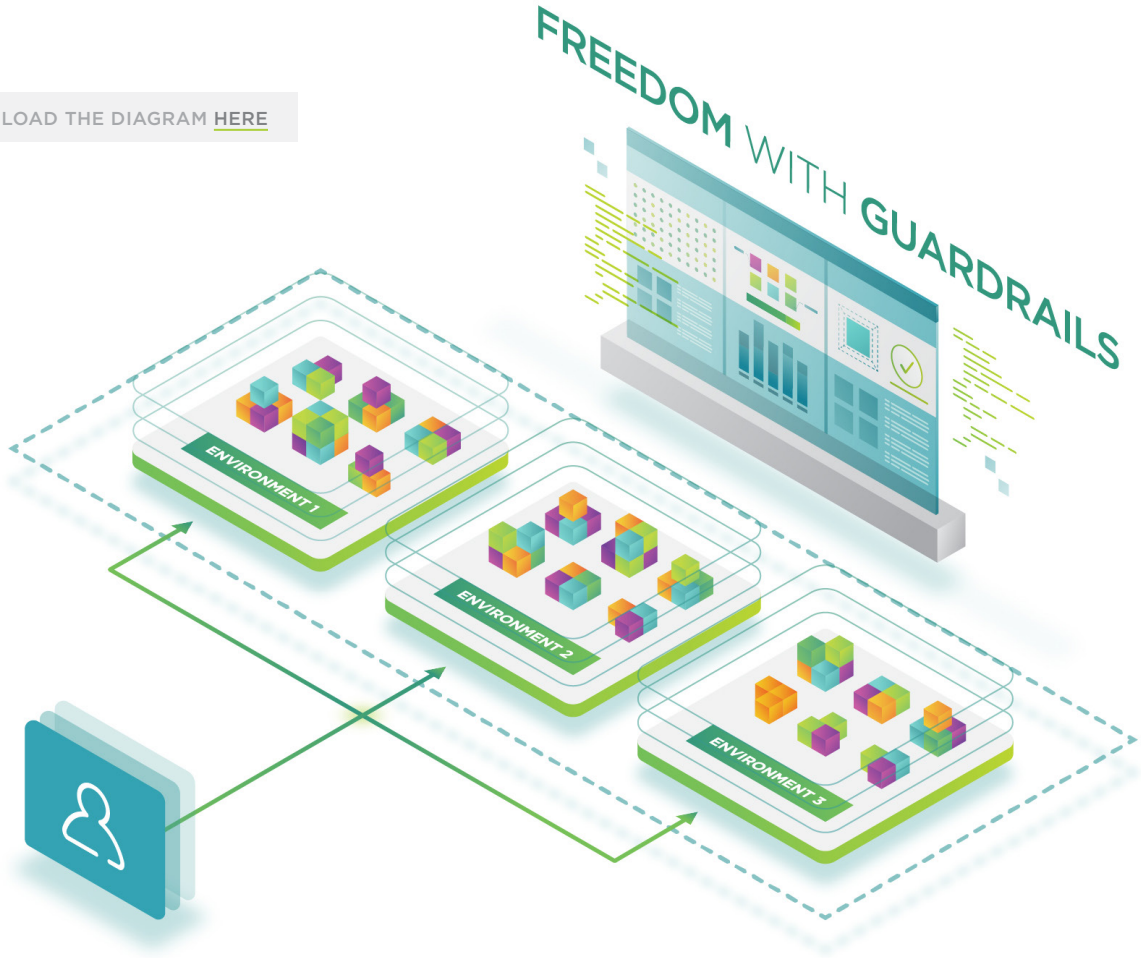
In the example deployment below, the end users are deploying cloud resources directly to the cloud environment, within the confines of a set of guardrails implemented by a wider platform or security team.

These guardrails are deployed in the form of access control boundaries, and end users are free to use any tool or capability to provision resources to deploy the workloads.

Continuous assurance is implemented through a mixture of cloud platform native or third-party tools that monitor the state of the resources by querying the cloud platform APIs and triggering notifications for non-compliant resources.

In more advanced implementations, these notifications may invoke automatic remediation actions.

DOWNLOAD THE DIAGRAM HERE



This deployment pattern exhibits the following characteristics and items for consideration during evaluation:

TARGET OPERATING MODEL

- Due to the direct access to the public cloud platform, there is a risk that new services can be consumed without establishing organisational controls enforced through a governance framework
- It is important to consider the potential for tooling or method sprawl, as teams may choose different third party offerings based on individual opinions or personal agendas

END USER EXPERIENCE

- This model provides the highest level of flexibility for development teams, allowing them to mix and match tooling that plays to the best practices of each service they leverage to run their applications
- As the cloud providers release new services and features, the developers are able to immediately start using them without a centralised team having to review them and implement them into a centralised library of patterns or mandated tooling

SECURITY AND COMPLIANCE

- Time will need to be spent on the development of external governance rules and checks, or the procurement of licences for a third party tool
- Depending on the maturity of the cloud providers' access controls, the services that implement the guardrails may not have the ability to impose controls and restrictions on all available services or configuration attributes

ORGANISATIONAL CAPABILITIES

- Although model provides the highest level of flexibility for development teams, it may become challenging in organisations who are still familiarising themselves with public cloud platforms as the providers continue to release new services at an accelerated pace

Tools and technologies that aid in implementing this cloud deployment pattern include:

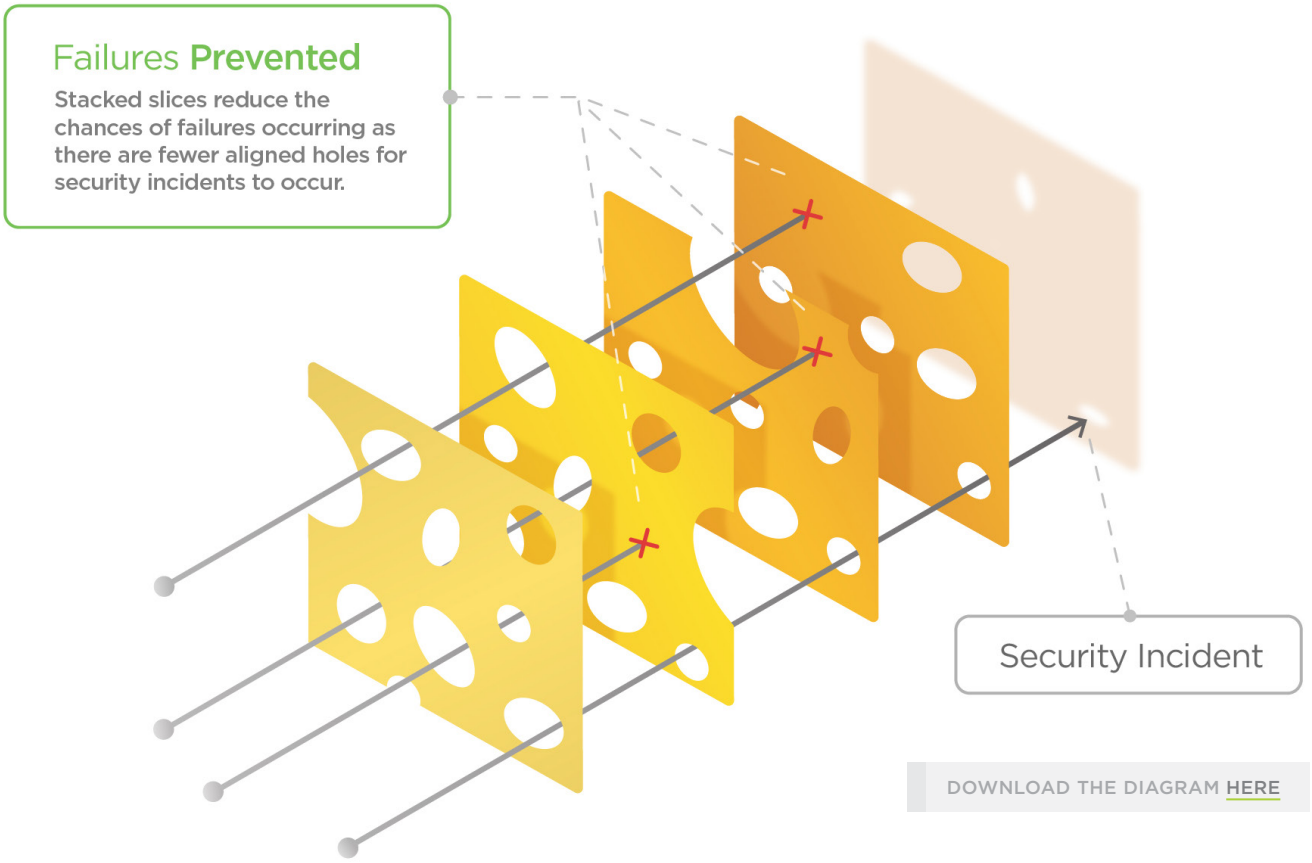
- Cloud Conformity
- Cloud Custodian
- Prisma Cloud
- Prowler
- Scout Suite
- Dome9
- AWS Managed Config Rules
- AWS Managed Conformance Packs

Implementing Control Strategy

In the realm of software development, the **Swiss cheese analogy** is often used where an organisation's defences against failure are modelled as a series of barriers, represented by slices of cheese.

The holes in the slices represent weaknesses in individual parts of the system, and are continually varying in size and position across the slices. The system produces failures when a hole in each slice momentarily aligns.

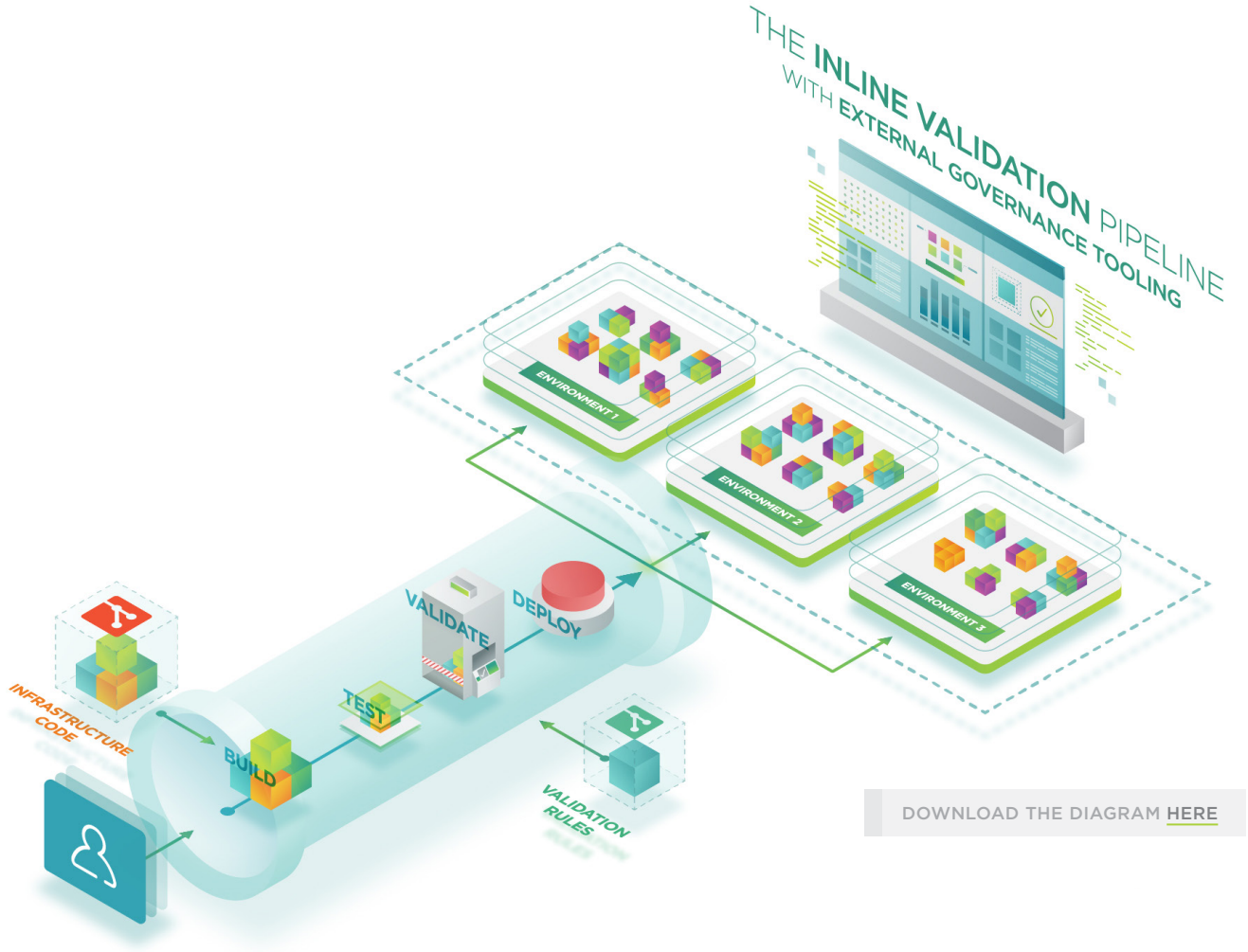
To reduce the chances of failures occurring, a software team may introduce additional slices of cheese (in the form of automated tests and checks) to further reduce the likelihood of bugs passing through to the final product, where they are most expensive to fix.



DOWNLOAD THE DIAGRAM HERE

In the same way that development teams introduce additional automated tests for software products, the cloud deployment patterns can be overlaid with the goal of providing greater visibility and governance over the cloud environments.

In the below example, we demonstrate how the Inline Validation Pipeline model can be combined with external governance tools and guardrails leveraged from the Freedom with Guardrails model.



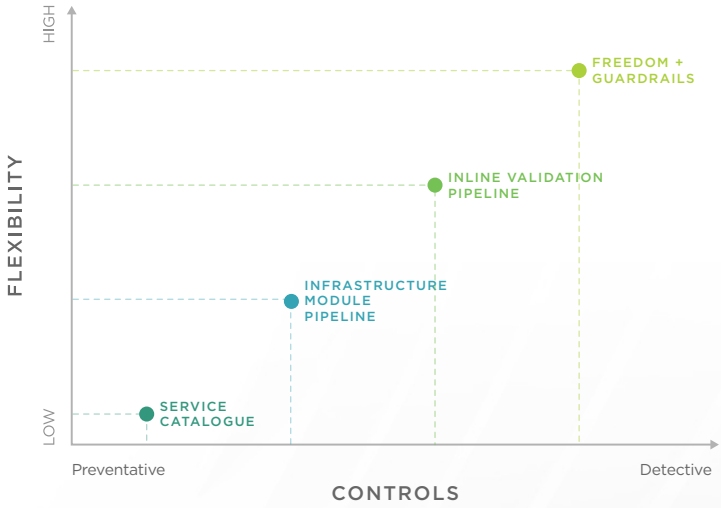
DOWNLOAD THE DIAGRAM HERE

Final Thoughts

Selecting a suitable Cloud Deployment Pattern is a foundational decision to enabling successful cloud adoption at scale. Organisations often start with a more prescriptive and preventative approach as they commence their journey, evolving to more flexible and detective models as they increase their confidence, capabilities and organisational awareness of the cloud security best practices and well architected principles.

When planning your deployment strategy, we encourage you to introduce the flexibility and controls objectives into your discussions, keeping in mind the organisational capabilities and the target operating model for the cloud environments.

It is also important to remember that as an organisation grows, it will likely seek to adopt additional deployment patterns to accommodate new requirements, evolving operating models and technical maturity.



Implementing a Cloud Deployment Pattern that meets the control objectives of your organisation is an important decision which needs to be revisited continuously as your organisation matures its consumption of cloud services.

Making the right, well informed decisions backed by knowledge of your organisation will accelerate cloud adoption, maintaining the right balance of flexibility and controls appropriate for your environment, while reducing friction and adoption hurdles for the end users.

Authors

The foundations of this paper were originally constructed by Sourced Consultants **Melody Huang** and **Keiran Sweet**, however this content represents the combined expertise of our peers.

We would like to especially call out the technical and creative input of **Yuri Litvinov**, **Salma Datenis**, **Pedram Sanayei** and **Jerri-anne Yap**. Without their assistance, this publication would not be possible.

References

Amazon Web Services (AWS)

Set Up a CI/CD Pipeline on AWS

AWS Service Catalog - Getting Started

Leveraging AWS CloudFormation to create an immutable infrastructure network at Nubank

AWS re:Invent 2018: Streamlining Application Development with AWS Service Catalog (DEV328)

AWS re:Invent 2017: Culture Shift: How to Move a Global Financial Services Organization (FSV308)

AWS re:Invent 2017: Cloud Adoption in Regulated Financial Services (SID328)

AWS re:Invent 2018: Cloud Custodian - OpenSource AWS Security & Governance (DEM78)

Microsoft

The Release Pipeline Model - Microsoft

Azure Resource Manager Templates

Static Analysis for Example Azure Data Lake Gen 2 Implementation

Static Code Analysis for ARM Templates?

Unlocking Azure with Puppet Enterprise (Puppetconf 2017)

Google Cloud Platform (GCP)

Laying the Groundwork: How to Build a Foundation in Google Cloud

Cloud Deployment Manager

Miscellaneous

Open Policy Agent (OPA)

Conftest

Order in a world of snowflakes (Puppetconf, 2015)

Accelerate Your Cloud Journey with Automated Controls and Governance (Future of Financial Services, 2020)

OPA Open Policy Agent Offering flexibility and security with policy as code (Cloudsec 2019)

Why Atlassian uses an internal PaaS to regulate AWS access

Effective Testing: The “Swiss Cheese” model

The Swiss Cheese Model (Wikipedia)

Killing Click Ops - What it is, why it’s a problem and how to avoid it

What Is GitOps and Why It Might Be The Next Big Thing for DevOps

Shifting Cloud Security Left - Scanning Infrastructure as Code for Security Issues

About **Sourced**

Sourced Group is a global cloud consultancy that helps enterprises make the most of cloud services with a focus on security, governance and compliance. With offices in **Australia, Canada, Singapore** and **Malaysia**, we provide professional services for securing, migrating and managing the cloud infrastructure of large enterprise clients in highly-regulated industries.

For more information, get in touch with us at **enquiries@sourcedgroup.com**

sourced
cloud at scale™

Sourced Group
sourcedgroup.com