appvia

Why Developer Self-Service is the key to cloud innovation

Why Developer Self-Service is the key to cloud innovation

Introduction to Developer Self-service

Developer self-service is a way of providing developers with the ability to provision the Cloud compute resources that they need for their application(s) to work - such as compute, databases, object storage and message queues - without needing to interact with a Platform, Cloud or DevOps engineer. Currently Cloud automation tooling is designed primarily for the Cloud Engineer, Platform Engineer or DevOps Engineer persona, resulting in a lot of time-bound activities sitting outside of the Development teams. This often hampers security visibility and awareness as well as introduces significant project delays.

> **“Developer self-service is a way of empowering business units to deliver their applications in the Cloud without needing to depend on specialist skills inside of the project team”**

Developer self-service is a way of empowering business units to deliver their applications in the Cloud without requiring specialist skill-sets to be present within the project team.

What exactly is developer self-service?

Developer self-service is a way to optimize the relationship between the Cloud and developers inside of the business units.

Cloud compute resources should be immediately available to applications creating a fully functional, end-to-end business service that results in faster time to market, expedited application modernization and faster Cloud migrations.

Developers should be able to repeatedly provision services without being domain experts on the Cloud or the tooling surrounding the Cloud. The expected level of knowledge should remain close to the application itself, meaning that they only

The type of Cloud services they need i.e. a Database, object storage, machine learning etc.

The number of environments they need to validate the quality and feature capability produced in the product.

The standard business requirements - such as the region required for hosting or the level of service availability needed.

An understanding of how to debug their application, find logs and where to receive information about issues.

There will always be an element of up-skilling needed in regards to the technology choices being made to support software delivery but it is important to keep this to a minimum so that the focus is primarily on delivering business functionality quickly, securely and repeatedly.

What developer self-service is not

Current Cloud automation tooling is very DevOps persona centric and assumes a lot of domain expertise in regards to Cloud automation, Cloud best practice and Cloud security.

Shifting the tooling left to developers without reducing the required domain knowledge cannot be considered Developer self-service as it does not have the correct impact on the business. The amount of developer upskilling required on Cloud operations, Cloud security and DevOps tooling (on top of their existing responsibilities of software engineering) would cause significant delays in time to market and product innovation.

If you are considering shifting things left to Developers then there has to be considerable effort on improving the developer experience and reducing the level of Cloud and operations knowledge needed in order to deliver software inside the organization.

The marker of successful Developer self-service is the speed of which a developer can be onboarded inside of the organization and deliver an effective outcome and the amount of time and effort invested in order to achieve that.

What are the benefits of developer self-service?

Providing developers with the ability to easily serve what they need whenever they need it removes the need to scale DevOps, Platform or Cloud teams as demand for Cloud scales up within a business.

This has multiple benefits:

- **Reduced cost:** as fewer resources are needed to meet the demand of the business.
- **Reduced skills dependency:** no reliance on high volumes of specialist skills, which are difficult to recruit and retain.
- **Increased delivery speed:** as developers have no wait time.
- **Faster time to market:** due to increased delivery speed.
- **Shift-left principles:** moves visibility and control to the developers in the business units that own the data, development, vision, marketing, operational risk, timelines, users and budget.
- **Repeatability and consistency:** one way of working across your delivery teams.
- **Improved security oversight:** a clearly defined delivery model makes it clear where security checks & balances should be implemented.

When Developers are provided with more efficient ways of working their value to the business can becomes more focused on capitlizing quickly on markets, faster product feature releases and cost reduction by being able to do more with less within the Cloud, Platform or DevOps teams.

What are the cloud delivery issues?

With multiple Cloud vendors and the ever increasing number of Cloud services to choose from, current ways of working are making Cloud adoption increasingly complicated, resource intensive and time consuming for a lot of organizations.

The recent "State of Cloud Report 2022" from Flexera shows security and skills as the biggest challenges facing organizations.

Cloud delivery is hugely reliant on tools and the experience of Cloud engineers. The quality of the security implementations are predicated on the knowledge of the resources and the understanding of Cloud security best practices.

The current short supply of skills in the market has inevitably increased security risk as time pressures on constrained resources have resulted in reduced quality and a high risks of security breaches.

Top cloud challenges for all organizations:

Source: Flexera, state of cloud report 2022

Figure 1.

What is causing cloud delivery issues?

One of the main issues facing organizations is the lack of consistency in how teams are delivering in the Cloud. Most businesses attempt to drive consistency through people and tools, often leading to high costs, misconfigurations and a limited understanding of the risks introduced from project to project.

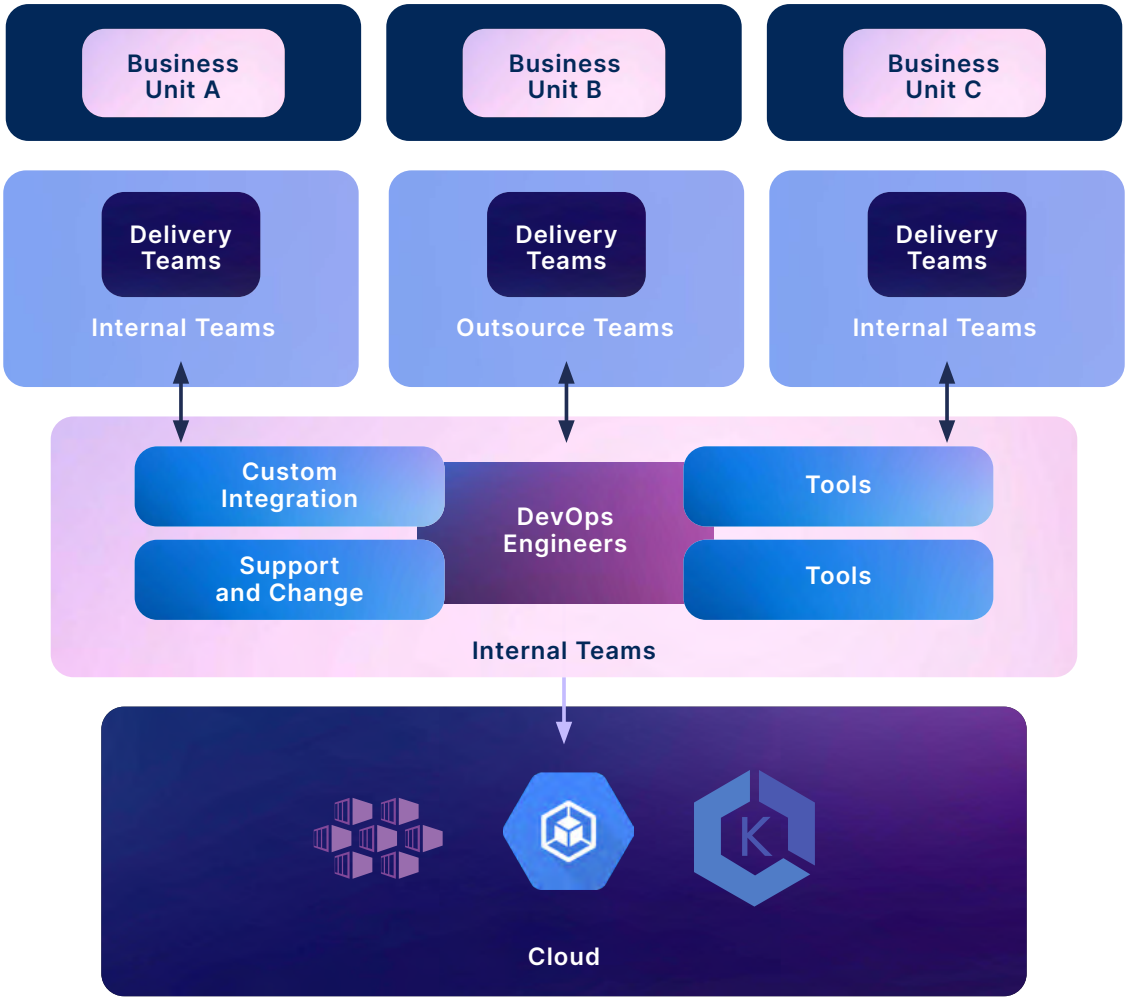

Figure 2.

How teams are delivering is equally as important as what they are delivering.

In the example above, you can see that there can be a mix of internal teams and outsourced teams all delivering in the Cloud. This mixed approach will inevitably cause deviations that can introduce problems at a significant cost to the business.

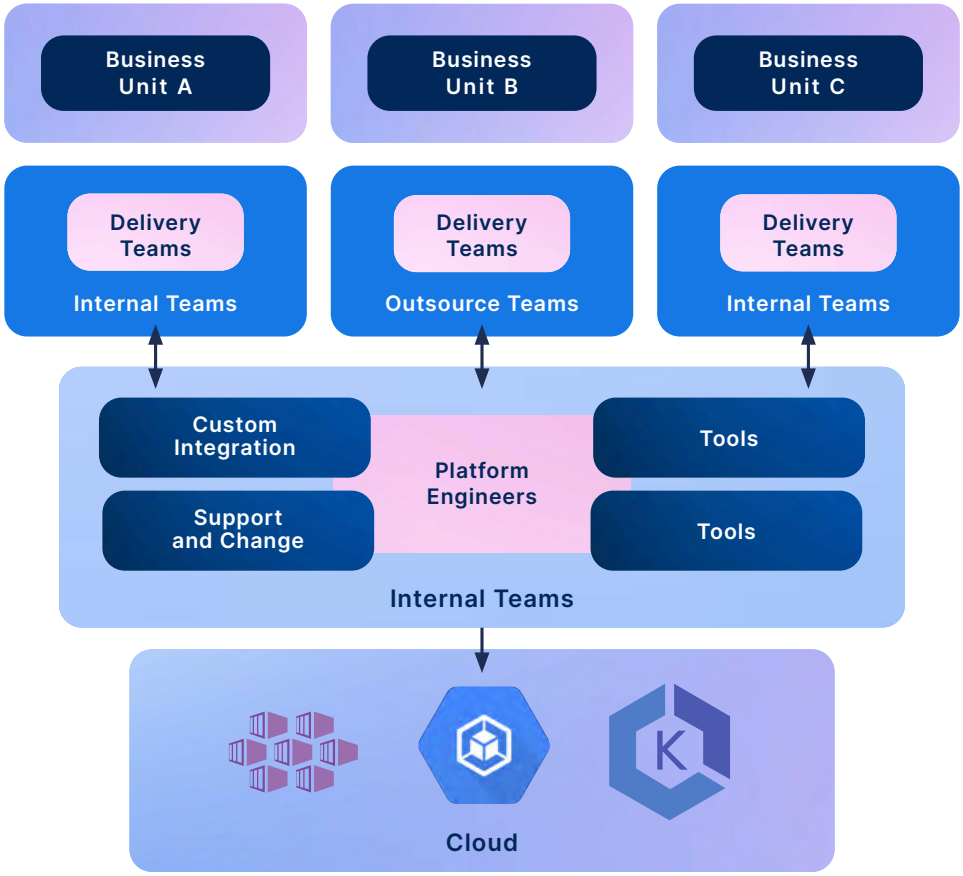

Figure 3.

Alternative delivery methods can provide more consistency, such as central Platform teams depicted below.

This has the added benefit of solving technical problems in one place and reducing costs across the business, however, without developer self-service capabilities the team will need to scale with the demand of the business.

What does it mean to devops?

Although DevOps methodologies are not related specifically to the Cloud, the Cloud makes it possible to achieve good automation through the exposure of programmatic interfaces (APIs).

Aligning to a fully automated process for all of the Cloud means that the right security checks and balances can be put in place throughout that process.

The automation my team uses improves the quality of our work
All agree responses

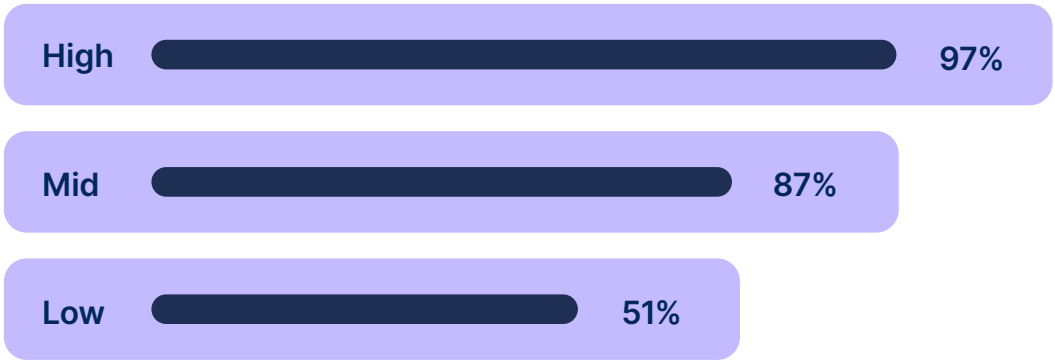

Figure 4.

In the 2021 state of DevOps report 97% of respondents agreed that automation improves the quality of the work of the team.

Continuous integration, testing, Cloud automation and security test automation are critical to reducing risk and improving quality in DevOps processes.

Automating cloud security

Automating Cloud delivery means that security checks can be put in place within that process.

Technologies such as checkov provide a huge number of security standards that can instantly be applied to Continuous Integration pipelines to validate the security of what is being produced by Cloud teams.

Enabling these checks makes security more visible to teams as they are iterating across the business and the results can be fed back into the CI pipeline checks to make sure teams fail fast and amend accordingly.

Having a central versioned security policy repository will provide consistent checks across your Cloud pipelines. This repository can be versioned and then consumed by others across the business, making it easier for policy and Cloud teams to collaborate and understand what checks and exemptions are in place.

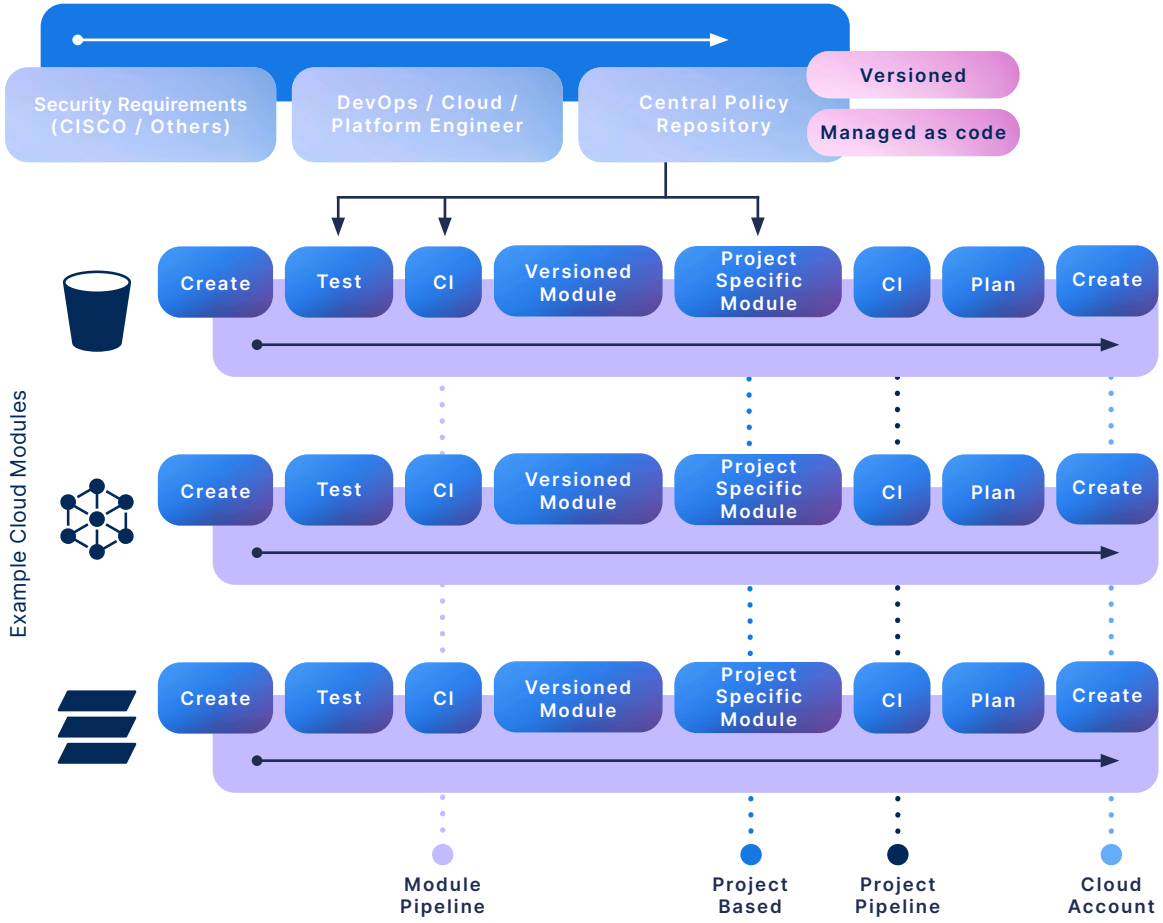

Figure 5.

The above diagram shows how these security checks can be centralized and where the checks should be happening:

Failing fast: Placing checks at test stage means the author can fail quickly and amend accordingly before submitting the changes

Central checks: Placing the same checks in your Continuous Integration pipelines will mean that changes made by others have checks performed in one place that is visible to the team

Project specific checks: Enforcing checks at the project level will make sure that any project specific configurations are also validated

Why not stick to devops and automation?

Although automating is critical to simplifying security and making sure there is a repeatable way of delivering, there are limitations and manual overheads even when best practices are followed.

These limitations have more to do with tooling in the industry and it being very DevOps centric and not necessarily business centric or developer centric.

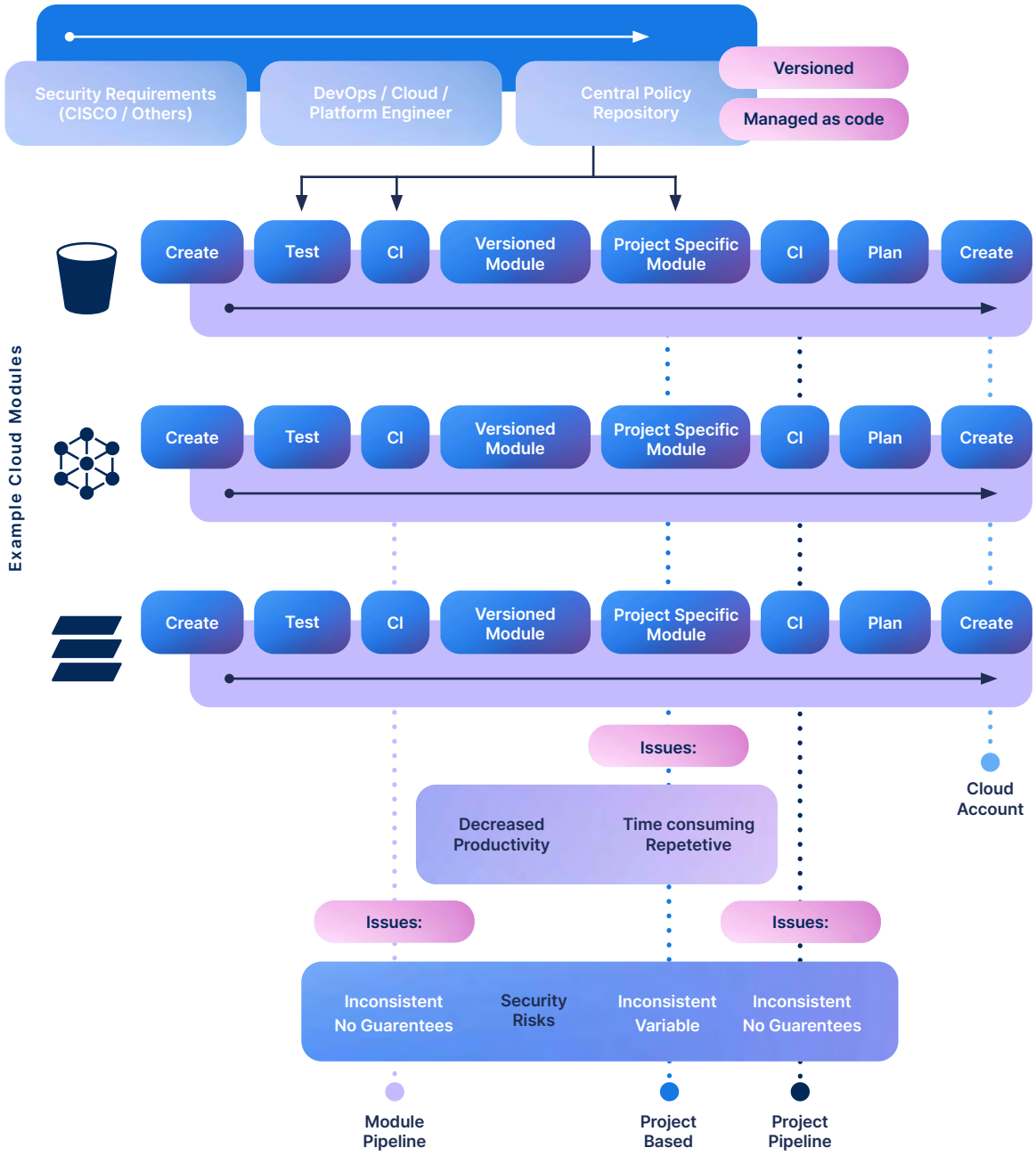

Figure 6.

Figure 6 highlights where the risks can still be, even with a firm automation process and a centralized set of security standards.

What causes the issues:

Manual continuous integration:
When CI pipelines are manually created by team members security steps can be missed, removed or commented out. Enforcing specific checks is difficult and is managed through a manual review process within the team.

Project specific parameters:
As each project has its own unique set of parameters (database name, object storage name or regions etc.), each Cloud module consumed needs to provide project specific information. This causes repository sprawl and reduces configuration oversight, resulting in security risks.

Runtime security:
Checks need to be enforced at runtime so that modules used to build the Cloud infrastructure are subject to the same security checks during the production of the Cloud modules.

Manual steps and slow time to market:
Each project needing its own unique set of Cloud configuration, pipelines, pipeline security checks and automation work can result in significant project delays and a need to scale the team up in relation to the demand.

What is the overall impact?

The typical impact of custom tool and process integration is an increase in security challenges, poor DORA metrics and a reduction in cloud-native approaches to delivery.

DevOps, Platform and Cloud engineers are so focused on attempting to manage and operate the Cloud delivery lifecycle that they are no longer spending time and energy on helping improve the quality of applications - be that in terms of design approach, developer education or training and adoption of Cloud best practices. This lack of developer education can result in poor cloud-native adoption processes, meaning applications are not designed and delivered in a way that sets them up for success in the future.

The metrics used to measure the effectiveness of a DevOps implementation are:

Mean time to recovery:	The amount of time it takes to recover from a system failure
Change failure rate:	How often a team's changes results in failures after new code releases
Deployment frequency:	The frequency of deployments to production
Lead time for changes:	The amount of time it takes from code change to a working deployment

These are useful metrics to consider throughout the software delivery process as they can be instrumental in understanding the quality of the applications being produced and as a marker of how Cloud native your organization is.

Additional metrics include:

Cloud delivery quality frequency:
How often are Cloud modules failing best practice tests

Security quality frequency:
How often are issues being raised against modules that are not complying to security standards

So how can I adopt developer self-service?

Developer self-service is still a maturing paradigm in the industry and, as with all technology, will mature and improve.

To understand how it can be achieved we have to break down the flow from figure 5 (pg. 12) further to understand who is responsible for what inside the chain and tackle the security issues and optimize the Cloud delivery outcome.

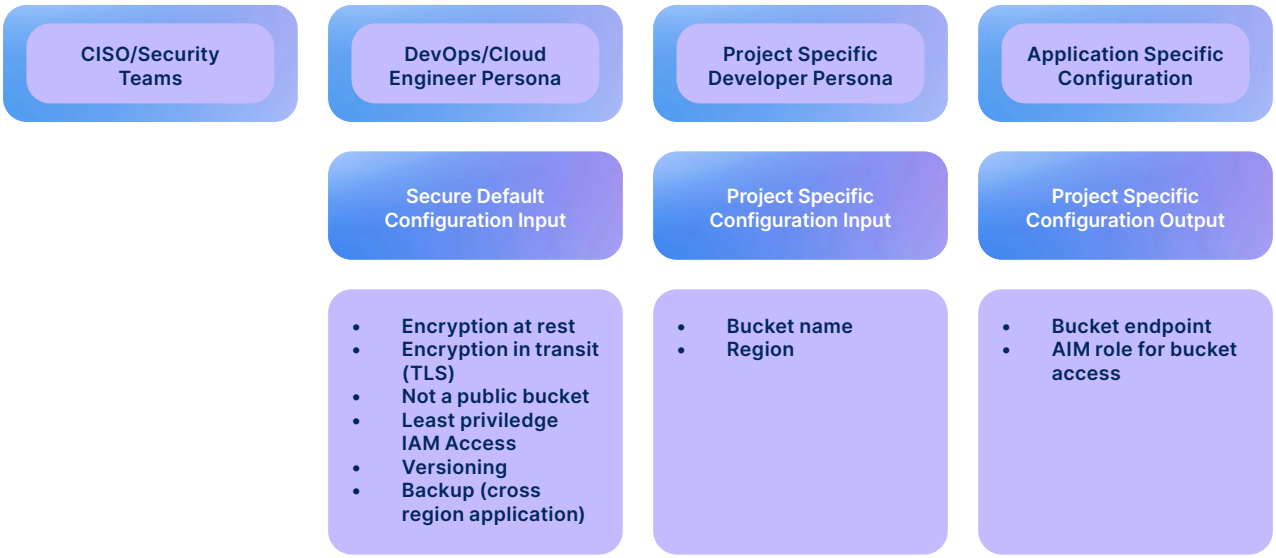

Figure 7.

If we were to take an example Cloud service such as a production S3 storage module then the only real information a developer would be required to provide would be the application specific details such as the region they want the bucket in, the bucket name and whether it is private or public. Some of these decisions (such as region or public / private buckets) may be enforced by security teams in the business.

If the DevOps, Cloud or Platform engineers are working with the security teams on the security requirements then the cloud modules can include default security configurations that the projects inherit.

If we shift left the project specific inputs to the actual development teams as shown in Figure 8 then we provide a simpler abstraction that is more tangible to the application delivery.

This means that developers can input known parameters about their applications without getting into the technical details of learning a DevOps tool and hence deliver faster and more securely.

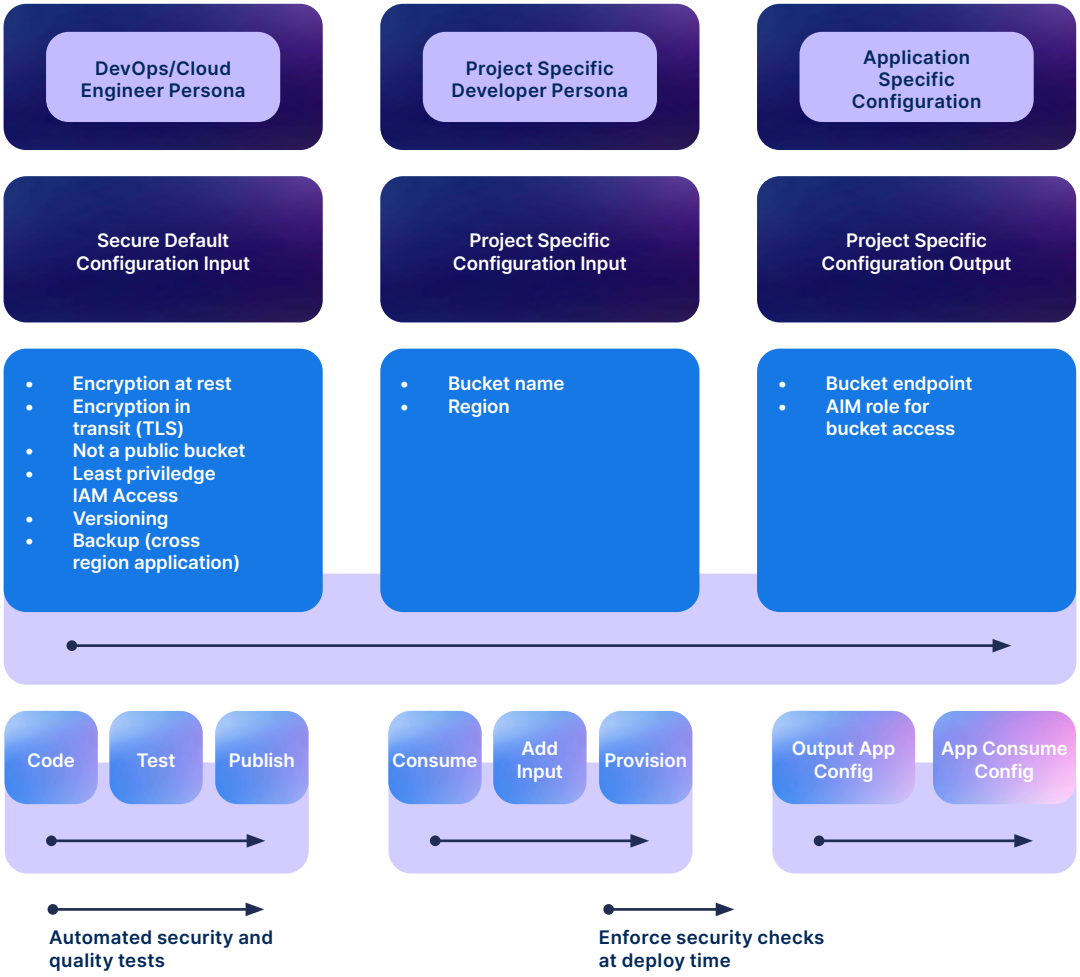

Figure 8.

Conclusion - Cloud delivery needs to change!

In order for businesses to meet Cloud demand in a scalable and repeatable way teams need to be presented with a secure, self-service Cloud capability that enables them to deliver quickly without compromising on security.

With Cloud automation technologies being primarily focused on the DevOps and Cloud Engineer persona and not the Developer, it has become a people and tool based approach rather than a commodity and product-centric approach, resulting in high delivery costs and the introduction of security risks.

Developer self-service is a way of empowering the business units to deliver and scale in the Cloud to unlock innovation and stay competitive.

> **“Developer self-service is a way of empowering the business units to deliver and scale in the Cloud to unlock innovation and stay competitive.”**