*Chad Ballay*

*Bellevue CYBR 450-342 | Advanced Cybersecurity Concepts*

*Enterprise Secret Storage*

Selling and Adopting

Table of Contents

[Abstract 2](#_Toc61844211)

[Introduction 3](#_Toc61844212)

[Scenario and Overview 3](#_Toc61844213)

[Reliance on Wisdom of the Crowd 3](#_Toc61844214)

[What Is In Scope? 4](#_Toc61844215)

[Reference Architecture 4](#_Toc61844216)

[Cloud Native Architecture 4](#_Toc61844217)

[Continuous Integration/Continuous Deployment Pipeline 5](#_Toc61844218)

[Devops and DevSecOps 5](#_Toc61844219)

[Driving Adoption 5](#_Toc61844220)

[Bibliography 6](#_Toc61844221)

# Abstract

Guarding database credentials, tokens, passwords and other secrets becomes a non-trivial risk when scaling above a single developer. Securely storing these secrets yet providing a low inertia way for authentication, authorization, and access to use these secrets is key to encouraging developer engagement.

In previous times, teams would have their own walled garden for their codebase and implementation. There would also be a high degree of geographic colocation as well as time in role tenure established. That is no longer the case. Code bases are shared publicly. Developers will get onboarded/offboarded more frequently. Finally, the practice of being able to pass a post-it note with the credentials no longer is viable.

This paper will focus on mechanisms to increase the adoption rate and decrease the time to functional. To do so I will cover several approaches for increasing insight into the environment, demonstrating value, and finally positioning to ensure compliance with the policy of adoption. I will use a generic reference architecture as illustrative, but the recommendations will be non-implementation specific.

# Introduction

Within the IT world, organizations have had secrets to manage since the beginning. The security posture was often of relying on impermeable security moats that allowed access outside of the proscribed team. This paradigm does not work any longer. Attackers are themselves embracing methodologies that enable their speed to market to rival cutting edge companies. The blending of private/public requires that the impermeable become permeable to allow for speed and efficiency. Code is now running on public hardware, infrastructure itself is implemented as code so it comes and goes with the execution of a few instructions. Developers are onboarded and then offboarded at a quicker and quicker pace. The geographic realities of remote workers and non-collocated offices have stymied the ability for in person credential sharing.

All of which has yielded a healthy adoption of creating a centralized secret management solution to handle authentication, authorization, and access. These solutions help to prevent operational blindness, product sprawl, and developer privilege abuse but only work to the degree they are adopted.

In the following sections I will lay out a reference architecture strictly for illustrative purposes to use in describing mechanisms to implement that will help better drive adoption. These mechanisms will be targeted for the reference architecture but will also be generic enough that with minimal cognitive load should translate to any modern architecture.

# Scenario and Overview

“Under the right circumstances, groups are remarkably intelligent.” – James Surowiecki

## Reliance on Wisdom of the Crowd

For selecting our reference architecture as well as guidance for selecting your own architecture to implement, I strongly recommend the “Wisdom of the Crowd” approach is taken. This approach assumes that best practices will emerge from a sufficiently large population. Good ideas will be noticed and socialized. Bad ideas will also be noticed and dropped. (Surowiecki, 2005)

This benefits the enterprise primarily in the reduction of cost for doing their own research. Not only in cost of computing resources but more importantly in utilizing sufficient numbers of man hours in evaluating the research to decide about proceeding further. In the case where an idea is researched but ultimately found to be lacking this cost has very little return on value.

A more generalize benefit is that by adopting broadly accepted toolsets and processes the underlying standardizations allow for a flexibility in the solution. A custom implementation of a highly tuned installation will show greater short term performance but would in the long term require the enterprise to limit any tools selected to a smaller subset of choices.

Finally, by using solutions that have a large userbase the enterprise enables themselves to select future employees from a much larger pool of potential applicants. In doing so this reduces the human constraint for ramping up capacity and throughput.

In short, outsourcing the research saves the enterprise resources, the results can be trusted due to the aligned interests of all involved, and the solution will likely enable more flexibility in resource and tool selection.

## What Is In Scope?

Secrets within this frame of reference is anything that you want to control securely store and tightly control access to tokens, passwords, certificates, and encryption keys for protecting machines, applications, and sensitive data. (Hashicorp, 2020) Generally, these are things where the enterprise would suffer greatly if an attacker could where to utilize in further their goals.

## Reference Architecture

For the focus of this paper, we are embracing a cloud native architecture that has implemented a modern CI/CD pipeline. DevOps principles are utilized to create an application that implements as much of the 12 Factor App framework. I’ll define those concepts but many of them are standards now.

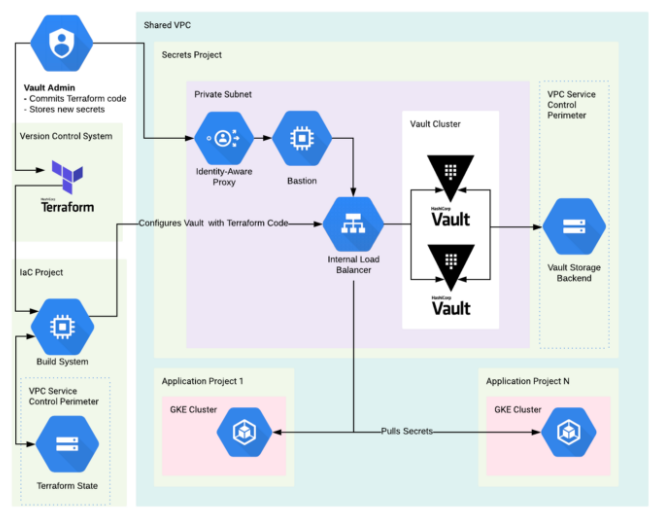


Figure - Sample Architectual Diagram - (Canty, 2019)

### Cloud Native Architecture

Our reference architecture will be using Google as the cloud provider. It will rely on network controls to enforce security borders. Infrastructure as Code will be used to dynamically build, utilize, and tear down infrastructure. What is actually running at the application is not needed for our purposes, only that some generic artifact is created and will be secure. The secrets and config values should fall in line with the goals of the 12 Factor App manifesto of removing them from source code and abstracting them out as environmental. (Wiggins, 2017)

### Continuous Integration/Continuous Deployment Pipeline

The transformation from source code to production artifact will have several steps and transformations to undergo. This collection of process and the tools used to implement them require setup effort but for this purpose we will treat it as a monolithic single process. The component steps generally can be summarized as follows:

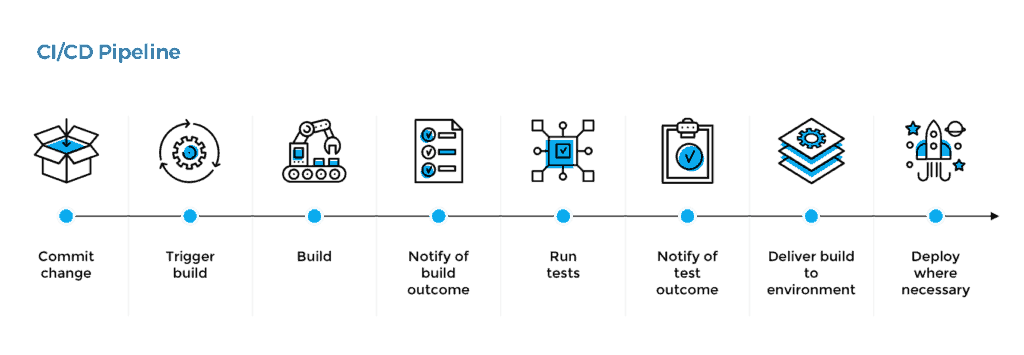


Figure - CI/CD pipeline abstraction - (Simmons, n.d.)

### Devops and DevSecOps

The original paradigm of DevOps has been expanded into sometimes being referred to as DevSecOps to highlight incorporating security in as well. The focus for our purposes is the quick release cycle that leverages automation of the software development lifecycle and production monitoring and deployment. (Forsgren, Humble, & Kim, 2018)

## Driving Adoption

Still trying to identify how to pivot from describing the environment to nuts and bolts driving uptake. I know I want to speak to how to monitor code, monitor commits, communicate back to developers when a secret is found in code, and how to track over time. Just can’t get it to work yet.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

# Bibliography

Canty, R. (2019, 10 22). *HashiCorp Vault and Terraform on Google Cloud — Security Best Practices*. Retrieved from Medium: https://medium.com/@jryancanty/hashicorp-vault-and-terraform-on-google-cloud-security-best-practices-3d94de86a3e9

Forsgren, N., Humble, J., & Kim, G. (2018). *Accelerate : the science behind DevOps : Building and Scaling High Performing Technology Organizations.* Portland: IT Revolution.

Hashicorp. (2020). *Unlocking the Cloud Operating Model: Security*. Retrieved from Hashicorp: https://www.datocms-assets.com/2885/1597082390-vault-com-security-whitepaper-v2-digital.pdf

Kovacs, E. (2017, 01 09). *"Truffle Hog" Tool Detects Secret Key Leaks on GitHub.* Retrieved from Security Week: https://www.securityweek.com/truffle-hog-tool-detects-secret-key-leaks-github

Marlow, P. (2019). *Finding Secrets in Source Code the DevOps Way.* Retrieved from SANS Reading Room: https://www.sans.org/reading-room/whitepapers/securecode/finding-secrets-source-code-devops-38985

Pidgeon, M. (2020, 01 23). *Conway’s Law and DevSecOps: How communication affects security*. Retrieved from The NextWeb: https://thenextweb.com/podium/2020/01/23/conways-law-and-devsecops-how-communication-affects-security/

Surowiecki, J. (2005). *The Wisdom of Crowds.* New York: Anchor Books.

Wiggins, A. (2017). Retrieved from The Twelve Factor App: https://12factor.net/

Zanelli, C. (2020, 10). *The Secret to Good Secrets Management for the Enterprise and Cloud* . Retrieved from Citihub Digital: https://citihub.com/insights/whitepapers/the-secret-to-good-secrets-management-for-the-enterprise-and-cloud/