DOPPLER

Redefining Secrets Management

Introducing the SecretOps Platform

WHITEPAPER

Contents

Executive Summary

The rise of multi-cloud has led many organizations to rethink their data-management strategies. Unfortunately, the ability to choose from different cloud vendors to select the best deployment offering for each type of application—though beneficial—has created fragmentation across environments.

The management of application secrets such as API keys and database credentials—comprising some of the most critical pieces of company data—are often not given sufficient attention despite the increasing trend toward security breaches and source code leaks. Meanwhile, teams continue to generate large quantities of secrets as they scale microservices and their infrastructures, causing issues such as duplicated secrets across deployments (e.g. an auth token for a private package or artifact repository) with no clear solution provided by traditional secrets managers.

This paper aims to provide a roadmap for developers, teams, and enterprises to successfully manage secrets in a multi-cloud world. Furthermore, it seeks to highlight the current inadequate approaches to secrets management in order to demonstrate why Doppler's vision of a SecretOps Platform provides the tools and workflows required to break out of traditional secrets silos and tame secret sprawl at scale.

What is Secrets Management?

To understand secrets management, we must first understand the purpose of secrets. Secrets provide applications with the required data for accessing privileged systems and services and come in various formats such as:

API keys	**SSH keys**	**Database credentials**
Authorization tokens	**TLS certificates**	

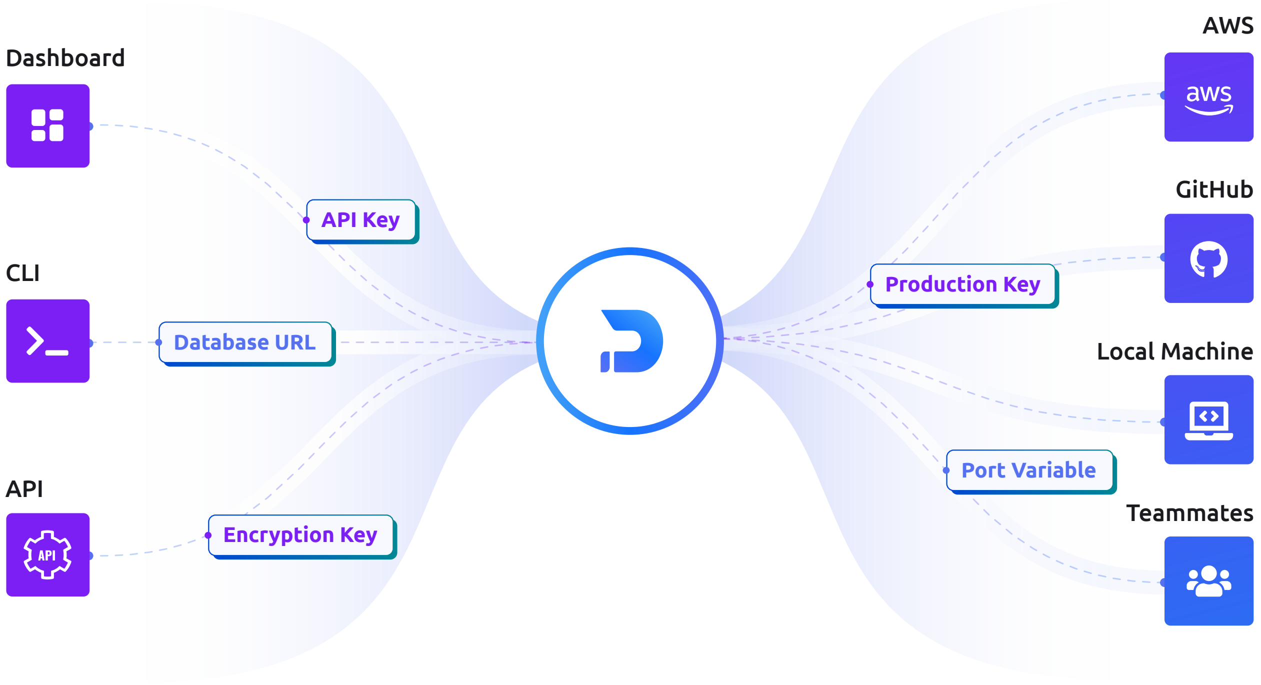
The vast majority of deployed applications require secrets in some form, and organizations must proactively protect them. Best practices for safeguarding secrets include scheduled secrets rotation and, when possible, ephemeral dynamic credentials. Frequent renewal of secrets limits the impact of breaches and is mandatory for compliance with security regulations laid down by HIPAA, SOC 2, and others.

However, such robust security measures are frequently unattainable for organizations lacking a dedicated Platform or DevSecOps team.

As organizations expand and new applications are created, the number of secrets continues to grow. This proliferation of company secrets makes it essential to apply the principle of least privilege when granting team members access to secrets. Further, when a team member leaves an organization, any credentials to which they had visibility should be considered for rotation as part of the offboarding process.

How Would a SecretOps Platform Help?

A SecretOps Platform simplifies and streamlines secrets management by introducing developer-focused workflows that don't exist in traditional secrets managers. Any developer on every team, regardless of the size of an organization, can therefore benefit from adopting a SecretOps Platform to ensure they adhere to security best practices while also improving developer productivity.

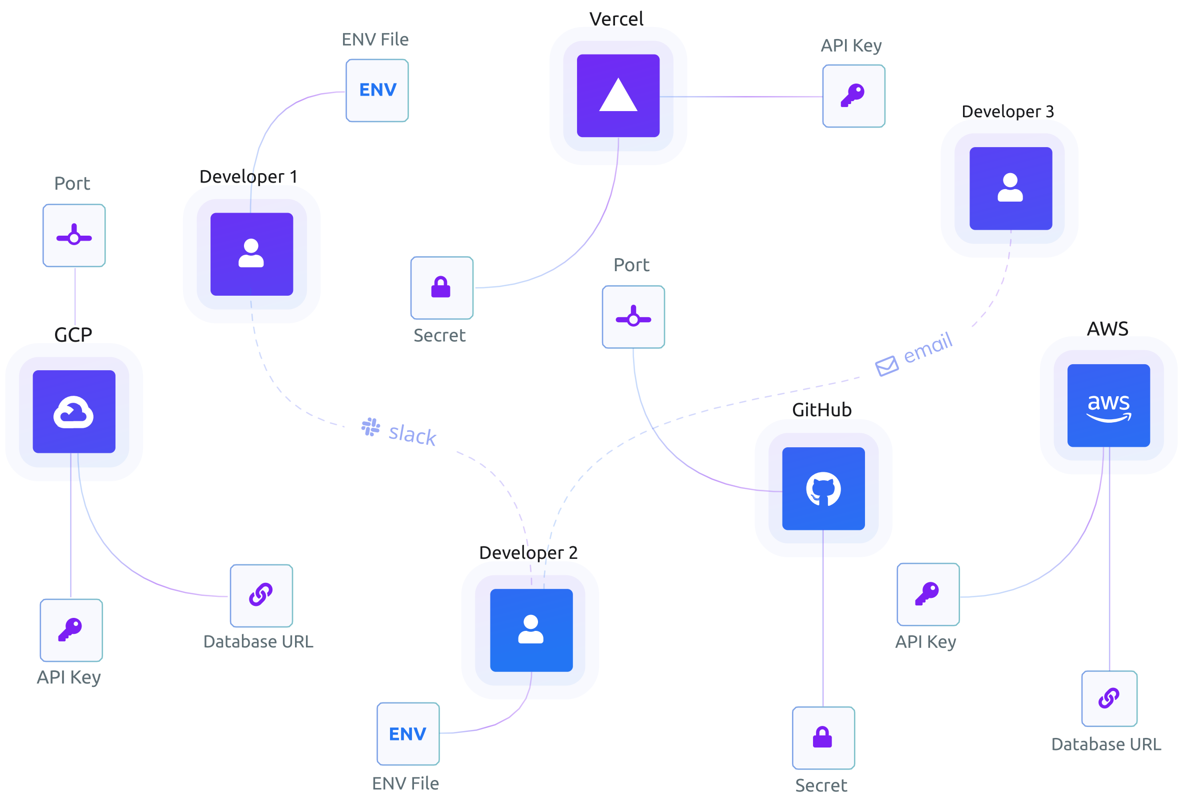


Secrets Considerations at Scale

Many developers fail to appreciate the scale and complexity of managing secrets at an organizational level.

Consider that the average enterprise now runs a *jaw-dropping 464 custom applications*. Microservices and the teams building them form the backbone of modern application development.

For most organizations, implementing a scalable, robust, and efficient SecretOps strategy is the key to taming secret sprawl. A SecretOps Platform provides a centralized source of truth for access control of secrets, organizing them according to microservice or project.



The Current State of Secrets Management

Currently, there are multiple ways in which companies manage their secrets:

- Hard-coded secrets
- Unencrypted secrets files
- Encrypted secrets files
- Traditional secrets managers
- Dynamic, ephemeral secrets

No system is perfect, and each of these avenues has unique drawbacks that must be considered.

Hard-Coded Secrets

Hard-coded secrets pose the most significant security risks. Anyone with access to the codebase also has visibility to the secrets contained therein. Should a malicious employee or external hacker acquire the codebase, secrets can be leaked anonymously, putting sensitive company data and systems at risk.

Scrubbing hard-coded secrets from an organization's many codebases is a complex and time-consuming task, even when automated code search and replacement tools are utilized.

Affected repositories can remain in developers' local environments in perpetuity, along with their commit history. This means the scrubbed secrets would still be accessible and must ultimately be rolled to ensure secrecy. Scrubbing hard-coded secrets is a task that's best avoided through proper management of secrets.

Migrating from hard-coded secrets to an organization wide secrets management solution is the best way to instantly improve application security posture by reducing the risk of secrets falling into the wrong hands.

Unencrypted Secrets Files

Unencrypted secrets files, such as .env files, are widely considered to be the most common method for supplying application secrets. While secrets files are more secure than hard-coded secrets as they are not included in the version-controlled codebase, they're a security risk due to being unencrypted at rest and prone to syntax errors as in most cases, they are managed manually.

The security risks are well documented, with bots constantly scanning the web for exposed .env files that exist within unprotected Amazon S3 buckets and public webroot folders. The history of S3 in particular, is littered with breaches stemming from access control policy misconfiguration:

- *PocketiNet exposed over 73GB of data, which included plain-text passwords and employee AWS secret keys.*
- *Viacom previously left numerous system credentials and data exposed in an open S3 bucket.*

This highlights the inherent risks of persisting secrets in an unencrypted format on disk, and we'll delve into more of their shortcomings later.

Encrypted Secrets Files

While using encrypted secrets files as part of a GitOps or sealed secrets workflow provides an additional layer of security, the management of encryption keys presents a new challenge. Even when keys reside within a protected centralized vault, careful access control is essential, and bespoke and often complex deployment solutions are required.

An alternative option some organizations embrace is the **"secrets manager"**—a dedicated system provided by a third-party service for storing and managing all secrets. They are perhaps one of the better solutions to date; their SaaS vendors possess deep expertise and handle the maintenance legwork.

However, these managers function mainly as Key-Value storage engines, and while offering top-tier security features, they often fall short in their actual management capabilities.

Traditional Secrets Managers

The market for secrets managers is well established, with major cloud vendors and HashiCorp Vault offering this service. However, they are designed for security engineers, not for developers.

Doppler's vision for a SecretOps Platform is to provide the secrets management experience that developers need and expect in 2022 and beyond.

Developers are constantly searching for new tools that make writing code more enjoyable, improve code quality, and speed up the shipment of features. Modern developer tools have raised the bar with slick, polished, and intuitive experiences from the start. But to win the hearts and minds of developers, it's not enough for a tool to just get the job done. The tool must demonstrate immediate value, and gain widespread adoption with minimal friction for developers.

Whereas traditional secrets managers provide a sufficient storage and encryption service to meet secrets governance and compliance requirements, their focus is limited to those requirements.

Consequently, they offer limited options for structuring secrets in the way a developer expects for an application. They were primarily designed to provide access to secrets for off-the-shelf applications—not to internal software development teams building and deploying applications.

A SecretOps Platform seamlessly integrates the tools, technologies, and services developers use to write, build, test, and deploy applications. Doppler seeks to do for secrets management what GitHub did for Git repositories—build a flexible and developer-centric platform with tools and automation workflows, combined with a beautiful and full-featured UI, and paired with a storage engine that meets the security of traditional secrets managers.

Secure secrets storage alone is no longer enough; developers will only embrace secure secrets management practices that are easier than insecure alternatives such as .env files.

Multi-Cloud, Vendor Lock-in, and Siloed Secrets

There's no doubt that the most intuitive tools are often the most useful. But, unfortunately, the secrets managers widely available to developers generally aren't user-friendly. They're not made for developer workflows or power users who want to accomplish the most in the least amount of time.

A significant advantage of the multi-cloud transition has been the flexibility to choose your preferred solutions, mix and match providers, and manage data and services based on the specific needs of each application.

Nevertheless, vendor lock-in persists since most secrets managers support a minimal number of integrations.

For example, when deploying an application to AWS, it might make sense to spend a significant amount of time integrating with AWS Secrets Manager. But what if running the same application on Azure becomes a better option in the future? The effort needed to migrate all of your applications and infrastructure to use Azure Key Vault may prove prohibitive. In short, your secrets management solution should never prevent you from choosing the best infrastructure by locking you in to your current cloud provider.

The reality of multi-cloud means breaking out of siloed secrets storage is a must—not a nice to have.

Our experience working with customers revealed that secrets tend to be sprawled across multiple secrets managers, such as Azure Key Vault and GCP Secrets Manager, and in the built-in environment variable stores on platforms such as Cloudflare Workers, Vercel, and Heroku.

Instead of transitioning or copying secrets between sources—risking data loss, human error, and potential downtime—choose a platform that's vendor agnostic and automatically syncs secrets to wherever applications access them.

The single biggest obstacle to companies efficiently and effectively managing their secrets in a multi-cloud world is their lack of knowledge about the benefits of a SecretOps Platform and what they should expect from one.

Consider what Uber did for the transportation industry. Uber's hyper-growth was not the result of clever marketing or positioning but the result of them raising the bar for what customers could and should expect from a transportation service.

It's our view that a similar revolution is taking place for secrets management tooling and workflows, which is what we'll spend the rest of this document discussing.

The Importance of Productivity

We've discussed why secrets management is essential to get right (preferably the first time around) and how secrets stored in unencrypted plain text formats such as .env files are inadequate in terms of productivity and security.

While traditional secrets managers have ensured better security, developers are forced to manually update key-value pairs in a limited UI on top of an encrypted data store. Developers want something better, something that solves the problems they face and meets their needs.

Developers treasure usability and productivity and wish to have similar features they've become accustomed to when shipping code safely and at scale, such as pull requests. The collaborative aspect of pull requests has completely changed how code is reviewed, checked, tested, and deployed. So why not take inspiration from a proven model such as pull requests and apply it to managing secrets?

Why can't alterations to secrets from a central source of truth be pushed immediately to other secrets managers or native environment variable stores in their hosting environment? GitHub Actions, Terraform, and Infrastructure as Code generally made the dream of reliable and automated builds and deployments a reality. But, until now, there has been no automation for managing secrets.

If a coding error causes a deployment to fail, we can easily roll it back in seconds thanks to Git and automated deployments. Why would we expect any less for secrets and configuration changes?

Current antiquated methods will likely force you to implement a manual and time-consuming rollback process. Ideally, what should take just seconds often requires developers to create tickets for their DevSecOps counterparts or struggle with confusing CLIs. Further, response times can vary widely depending upon team structures and time zone differences—a significant issue considering how quickly secrets management tasks must be attended to.

The massive shift to remote working has led to more automation across the board, which in turn is affecting teams' expectations for how secrets should be managed. Even when things work as expected, accessing secrets from traditional secrets managers can be complicated.

Secrets maintained within AWS Secrets Manager and GCP Secret Manager may be quite disorganized unless carefully considered best practices are uniformly applied throughout an organization. Different path-structuring rules, naming conventions, and the peculiarities of each platform greatly complicate this task, tempting even the most dedicated developers to give up and settle for the use of .env files, for example, as the best solution they can hope for.

At Doppler, we repeatedly heard these concerns from customers, and we realized that a better solution must exist. A solution designed from the ground up for multi-cloud deployments.

Motivated by our desire to raise the bar and better serve our customers through innovation, we created our design and vision of a SecretOps Platform that surpasses the traditional secrets management model. We wanted to improve every aspect of secrets management so our industry as a whole could enjoy greater productivity without compromising security standards and compliance.

A Secrets Platform for Modern Software

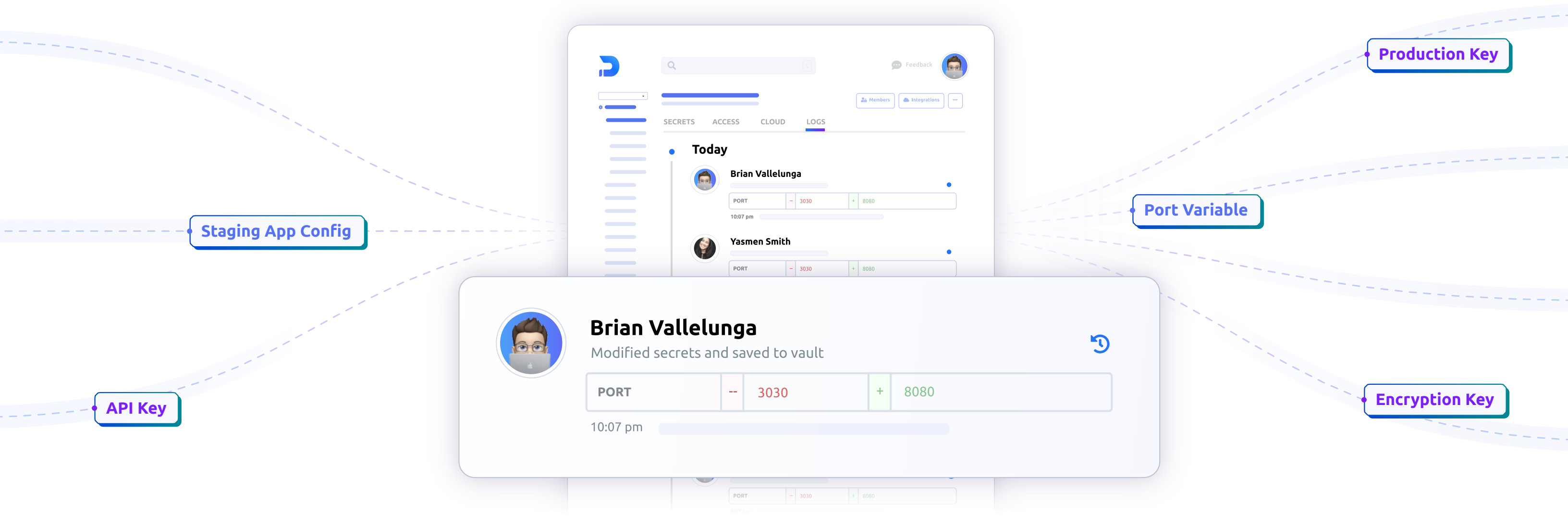
While helpful, even dedicated secrets solutions are limited in their scope and offer little opportunity for developers to take complete control of their secrets where they need to be used. What developers crave is having true management capabilities that align with the realities of the systems they use daily.

Hence, our solution, the **SecretOps Platform**. While secrets management may not be the most glamorous aspect of building software, our goal has always been to transform it into something teams would choose and want to use. For example, developers target rapid rollout, while security teams prefer traditional products geared toward their concerns.

A SecretOps Platform is designed to eliminate this disconnect between developers and DevSecOps teams, who often clash while balancing deployment speed against ecosystem security. This agility is increasingly critical in a multi-cloud context where microservices with smaller codebases are the norm. By including both developers and DevSecOps-friendly features, Doppler can be the single source of truth of secrets for every application, project, environment, and off-the-shelf software.

Doppler's SecretOps Platform has empowered developers by showing new secrets management practices that expose the flawed and insecure methods used currently, such as using .env files.

The Doppler Difference



Our SecretOps Platform contains every feature teams need to securely store and manage encrypted secrets—from start-ups to enterprise organizations. Our SecretOps Platform was designed from the ground up, adhering to security best practices for compliance and access control, but with a superior user experience compared to existing secrets managers. This ensures rapid adoption, usually with no application code changes required as integration occurs at the infrastructure layer.

Doppler is a fully managed service with an uptime guarantee of 99.95% platform-wide. Doppler also integrates with external secrets managers as part of the underlying tokenization engine, offering encryption key flexibility to enterprise customers. Additionally, we respect compliance guidelines and value transparency with every possible aspect of our service management practices, with logging and auditing functionality available through our dashboard or streamed to an external log ingestion service.

So what core features differentiate a SecretOps Platform from traditional secrets managers?

This is best explained by the design principles guiding our vision of a SecretOps Platform:

- ▸ *Doppler is multi-cloud ready, enabling you to integrate with the external platforms, secrets managers, and CI/CD tools.*
- ▸ *Doppler supports language-agnostic environmental variables for secure secrets injection at runtime—reducing friction and eliminating the need for SDKs and .env files.*
- ▸ *Doppler acts as a single source of truth for every aspect of your secrets management needs with developer-specific workflows that go beyond simple key-value storage.*

The management layer empowers developers to modify and securely sync or inject secrets as required, with the dashboard supporting the common tasks of creation, rotation, and revocation. The CLI offers a programmatic interface for those that prefer to stay in the terminal.

The Doppler dashboard encourages collaboration between developers, DevOps, and Security Engineers, helping teams avoid confusing Slack threads and other frustrating communication channels that often cause unnecessary delays to the deployment process or accidental leaks. For example, it's a little-known fact that Slack messages aren't as private as you may think.

If a change needs to be reversed instantly, initiating the rollback of a secret value is done in a single click using Doppler's activity log with access controls, ensuring only authorized users can make critical changes.

Doppler meets enterprises' fine-grained access control requirements, so only trusted administrators can alter secrets—which are managed according to individual user permissions, group-policy permissions, or organizational units.

Doppler offers four levels of access:

▶ **Owner**	⟶	*Can access everything within the Doppler workplace*
▶ **Admin**	⟶	*Can access all projects, without management access to the team, settings, and billing dashboards*
▶ **Collaborator**	⟶	*Requires access to Projects and environments be granted by an admin*
▶ **Viewer**	⟶	*Same permissions as Collaborator, but with read-only access*

Users can be authenticated using email and password with MFA. SAML and SCIM 2.0 are supported for dynamic user provisioning at an enterprise scale, making managing workplaces of any size effortless. Our internal administration console and data-protection processes also prevent any Doppler employee from ever being able to access your secrets.

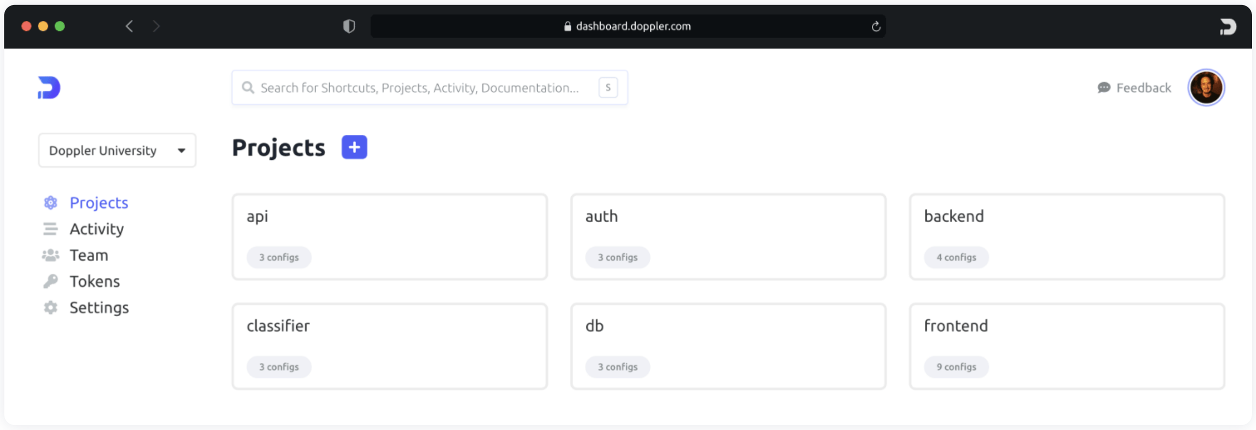
Where Management Meets Automation

Our **automation layer** is perhaps Doppler's most significant contribution to secrets management. Automation is the major missing piece in traditional secrets managers and certainly lacking from the error-prone process of manually keeping .env files in sync across multiple environments.

A single source of truth that eliminates inefficient communication methods for secret updates such as Slack or email is pivotal to streamlined secrets management and preventing easily avoided and costly mistakes.

Doppler provides an experience that makes updating secrets feel similar to changing code. A SecretOps Platform includes a set of developer-centric tools, automation workflows, and a fully-featured UI that feels like a logical extension of already familiar concepts for writing and shipping code.

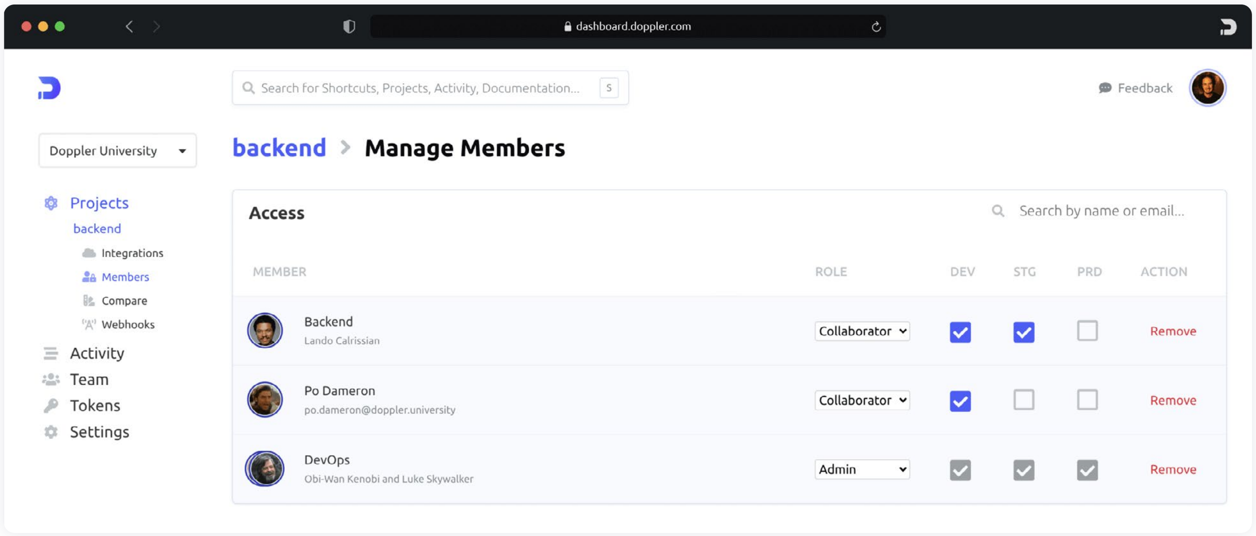
Doppler organizes secrets by application or microservice into **Projects**, each with a list of user-customizable environments.



PROJECTS EXPLORER

Enjoy an organized view of your active secrets using the Dashboard, and add new secrets as needed—all within one location.

Secrets access is defined on a per-project and per-environment basis. For example, backend developers can be granted access to the development and staging environments, but only DevSecOps has access to production secrets. The best part? Everything is managed visually via the Doppler dashboard in a few clicks with no complex policy language to learn.



backend > Manage Members

Access

Search by name or email...

MEMBER	ROLE	DEV	STG	PRD	ACTION
Backend Lando Calrissian	Collaborator	✓	✓	☐	Remove
Po Dameron po.dameron@doppler.university	Collaborator	✓	☐	☐	Remove
DevOps Obi-Wan Kenobi and Luke Skywalker	Admin	✓	✓	✓	Remove

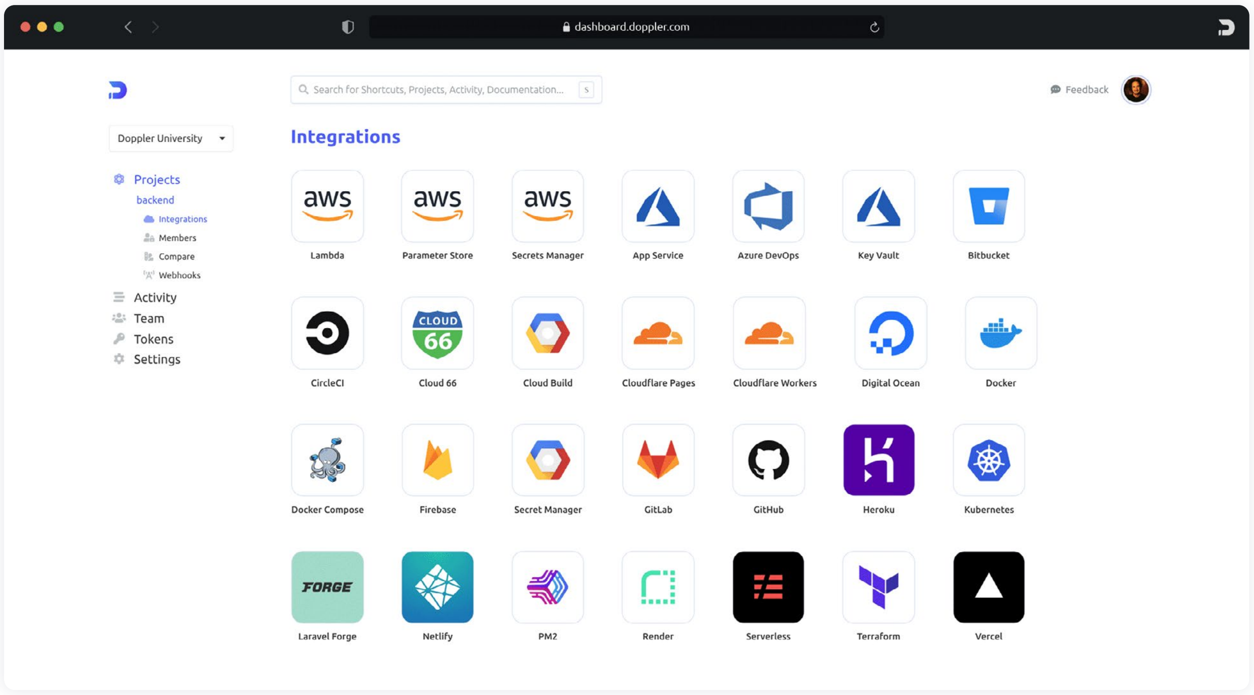
PROJECT ACCESS CONTROLS

One of the biggest differentiators between Doppler and traditional secrets managers is that Doppler positions itself as the centralized management solution and source of truth for storage without forcing applications to integrate with it directly in every instance. That's a bit of an abstract description, so let's get more concrete.

Doppler recognizes that many platforms such as GitHub, Azure, AWS, GCP, and Vercel have built-in secrets storage solutions. As a result, developers will likely prefer to access their secrets via one of these stores, given that they are native, secure, and easy to use. The fragmentation we see in deployment platforms is only going to increase. Therefore, the most effective way organizations can manage secret sprawl is not to restrict the flow but to control it through integrations that sync directly to the secret stores for each platform.

Consider an application with a Vercel-hosted frontend, AWS Lambda APIs, and a Kubernetes backend hosted on GCP. Doppler integrations enable instant and automatic secrets syncing on change and are configured at a Project's environment level. For example, staging and production secrets are synced to different application instances.

Effectively a hub-and-spoke model with a SecretOps Platform at the center, pushing secrets to providers as secrets change.



INTEGRATIONS CATALOG

But syncing secrets to third-party providers is only table stakes for Doppler, as a crucial piece that's missing from almost every current deployment solution is the automated reloading of applications when secrets change. Again, this is a prime example of a SecretOps Platform thinking of workflows DevOps teams need beyond simply storing and accessing secrets.

Just as GitHub helps teams write better software, not just host Git repositories, a SecretOps Platform helps teams manage secrets more thoughtfully and efficiently in an almost inconceivable manner compared with traditional secrets managers.

Doppler integrations can also offer automatic application redeployment when secrets change, either through the integration itself, such as the Doppler Kubernetes Operator, or using Doppler webhooks with platforms such as Netlify and Vercel.

Innovation in deployment automation has touched almost every aspect of software development over the last ten years. Still, very little has changed when it comes to managing secrets, mainly because we've all become numb to the pain of manually managing secrets in .env files or relying on complex secrets managers.

While a SecretOps Platform captures our vision for what secrets automation could be, the ultimate driver that will revolutionize our industry and bring about large-scale and lasting change is when developers expect the same level of automated efficiency from secrets management that they do from their existing infrastructure tooling.

Enterprise Capable Access Controls

Successfully applying the principle of least privilege at scale requires more than just fine-grained access controls. It's also crucial how quick, easy, and accurate it is to apply them.

Complexity in defining access control is not only a risk to implementing them incorrectly, but also to them being implemented at all. The considerable number of AWS S3 data breaches resulting from poorly implemented access policies demonstrates that as difficulty increases in applying access controls, security almost always decreases as a result.

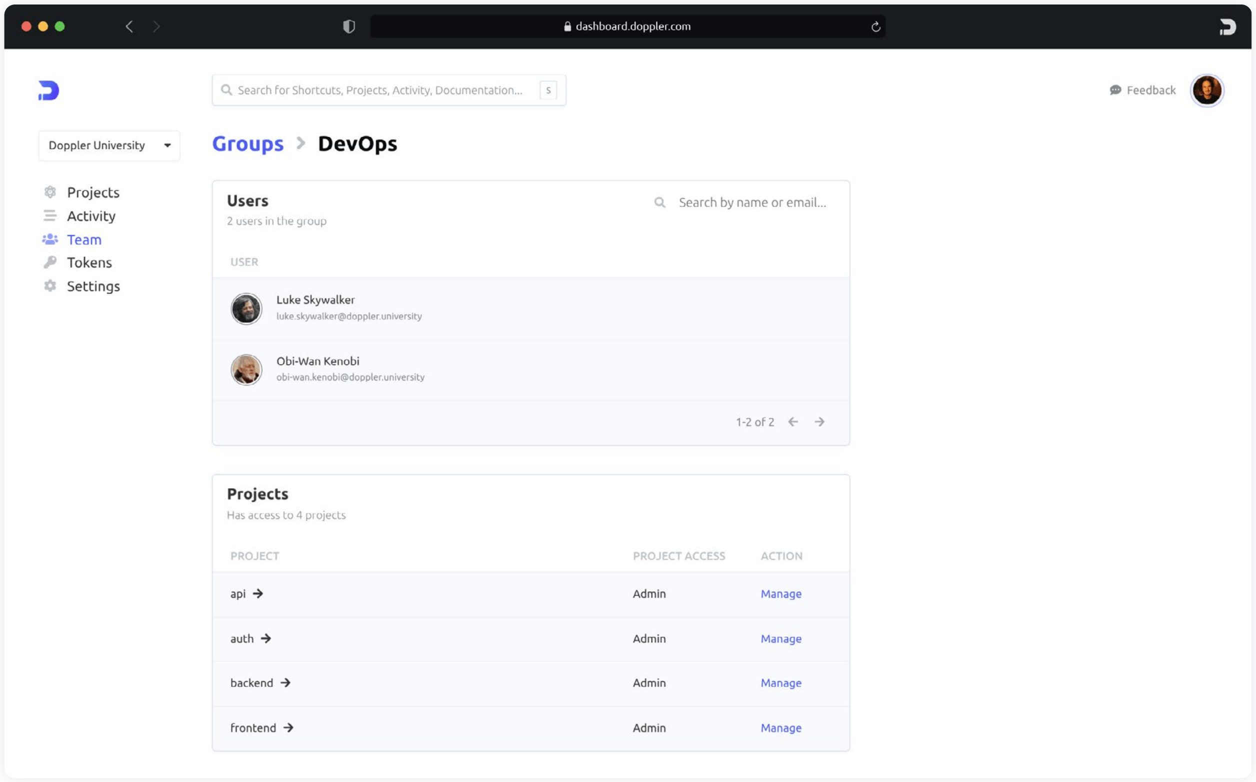
Traditional secrets managers check the box of "fine-grained access controls" using document-based access policies in JSON, YAML, or vendor-specific DSLs such as Hashicorp Configuration Language. Because traditional secrets managers only provide Key-Value storage, a secret path is defined in the key as a means of differentiating secrets belonging to different applications. Access policies are then applied to a portion of, or the entire key path, depending upon the use case.

For example, here is an overly simplistic example taken from HashiCorp Vault's policy documentation:

```
path "secret/foo" {
  capabilities = ["read"]
}
```

A real-world example would require substantially more rules such as secrets list, create, update, and delete permissions. While it's conceivable that this level of granularity may be necessary in some instances, it's a huge price to pay for the majority of more common and simple use cases.

By contrast, Doppler's team and user groups exemplify how we've built the required secrets access configuration tools right into our dashboard with no complex and confusing access policy syntax required.



Groups > DevOps

Users
2 users in the group

Search by name or email...

USER
Luke Skywalker luke.skywalker@doppler.university
Obi-Wan Kenobi obi-wan.kenobi@doppler.university

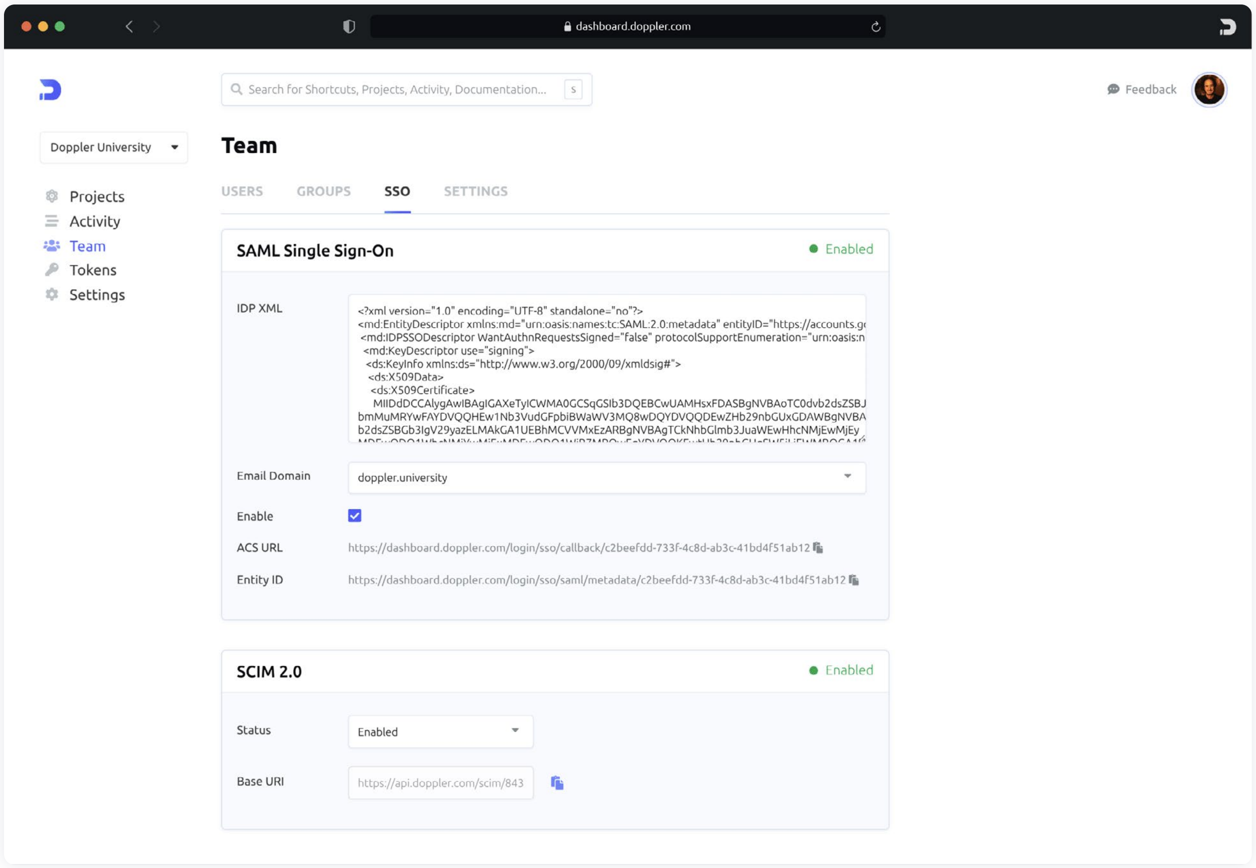
1-2 of 2

Projects
Has access to 4 projects

PROJECT	PROJECT ACCESS	ACTION
api →	Admin	Manage
auth →	Admin	Manage
backend →	Admin	Manage
frontend →	Admin	Manage

USER GROUP MANAGEMENT

Enterprise customers can dynamically provision Project member access using grouping rules from their Single Sign-On using provider using SCIM 2.0.



Team

SAML Single Sign-On ● Enabled

IDP XML

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata" entityID="https://accounts.g
 <md:IDPSSODescriptor WantAuthnRequestsSigned="false" protocolSupportEnumeration="urn:oasis:n
  <md:KeyDescriptor use="signing">
   <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:X509Data>
     <ds:X509Certificate>
      MIIDdDCCAlygAwIBAgIGAXeTyICWMA0GCSqGSIb3DQEBCwUAMHsxFDASBgNVBAoTC0dvb2dsZSBJ
bmMuMRYwFAYDVQQHEw1Nb3VudGFpbiBWaWV3MQ8wDQYDVQQDEwZHb29nbGUxGDAWBgNVBA
b2dsZSBGb3IgV29yazELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWEwHhcNMjEwMjEy
```

Email Domain: doppler.university

Enable: ✓

ACS URL: https://dashboard.doppler.com/login/sso/callback/c2beefdd-733f-4c8d-ab3c-41bd4f51ab12

Entity ID: https://dashboard.doppler.com/login/sso/saml/metadata/c2beefdd-733f-4c8d-ab3c-41bd4f51ab12

SCIM 2.0 ● Enabled

Status: Enabled

Base URI: https://api.doppler.com/scim/843

SAML & SCIM SETTINGS

Now let's cover how applications access secrets from a SecretOps Platform.

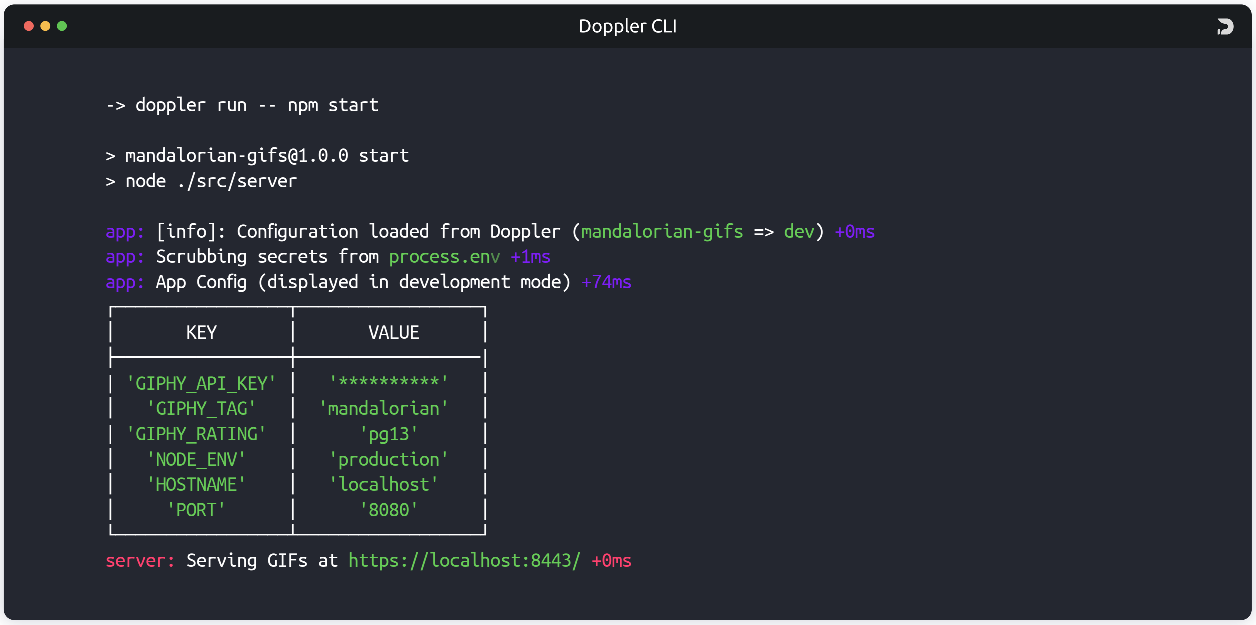
Accessing siloed secrets from a traditional secrets manager is often a significant source of frustration for developers because of three main reasons:

1. *Requirement of platform-specific SDKs tightly couples application code to a single vendor.*
2. *Complex to use SDKs negatively impacts developer productivity.*
3. *No native support for replicating cloud-based secrets manager access during local development.*

Doppler simplifies secrets access using either Service Tokens to provide read-only access to a specific Project and Config, or by using Doppler integrations that automatically sync secrets to external platforms—even other secrets managers.

Service Tokens provide a simple, secure, and flexible solution for accessing secrets in a wide range of scenarios:

- *Supplying secrets when provisioning Terraform resources*
- *Syncing secrets to Kubernetes deploymvents with auto-redeployment on secrets change*
- *Secrets injected as environment variables into the application process using the Doppler CLI*
- *Fetching secrets directly from application code using the Doppler API*
- *Injecting secrets as environment variables in CI/CD jobs such as the GitHub Doppler CLI install Action*


Doppler CLI

```
-> doppler run -- npm start

> mandalorian-gifs@1.0.0 start
> node ./src/server

app: [info]: Configuration loaded from Doppler (mandalorian-gifs => dev) +0ms
app: Scrubbing secrets from process.env +1ms
app: App Config (displayed in development mode) +74ms
┌─────────────────┬────────────────┐
│       KEY       │     VALUE      │
├─────────────────┼────────────────┤
│ 'GIPHY_API_KEY' │  '**********'  │
│   'GIPHY_TAG'   │ 'mandalorian'  │
│ 'GIPHY_RATING'  │     'pg13'     │
│   'NODE_ENV'    │  'production'  │
│   'HOSTNAME'    │  'localhost'   │
│     'PORT'      │     '8080'     │
└─────────────────┴────────────────┘
server: Serving GIFs at https://localhost:8443/ +0ms
```

DOPPLER CLI: SECRETS INJECTION INTO APPLICATIONS

Integrations sync updates from Doppler to existing secrets stores such as Azure Key Vault or AWS Secrets Manager, enabling customers to instantly benefit from unique Doppler features such as cross-project secrets referencing. This flexibility highlights a critical distinction between a SecretOps Platform and traditional secrets managers by allowing customers to choose how secrets are managed, stored, and accessed without being locked in by vendor-specific SDKs and secret silos.

Doppler also solves the problem of supplying secrets during development by allowing developers to install the Doppler CLI locally and access secrets from a project's development environment.

A SecretOps Platform must efficiently and effectively enable users, teams, and groups to manage access permissions to secrets at scale. Only then can organizations feel confident they are implementing the principle of least privilege that ensures application secrets remain that way and don't end up in the hands of attackers.

Summary

Traditional secrets managers were never built to support a multi-cloud deployment strategy or provide developers with the management features they need as part of the software development lifecycle and deployment process.

But we don't just need slightly better secrets managers. We need a SecretOps Platform. A platform that satisfies the requirements of both Developers and Security Engineers. Development teams and DevSecOps.

The level of complexity in using a secrets manager is not a badge of honor that indicates security at the highest level. A SecretOps Platform works with you to make your life easier and writing code more enjoyable while still meeting the highest security standards that enterprise organizations require.

Finally, we believe there's a moral imperative to securing customer data. Hackers are getting craftier and favoring infiltration and source code leakages over infrastructure attacks. Those secrets are the keys to a company's kingdom, and we believe a SecretOps Platform such as Doppler is a critical defensive component to minimizing secret leaks and breaches in the modern age.

Join the 9,000+ startups and enterprises that have adopted Doppler and now think differently about secrets management.

Doppler is free to get started or get your enterprise trial underway by jumping on a quick call with our team.

DOPPLER

Schedule a session with our team to discover how the enterprises of tomorrow are powered by our holistic secrets platform.

Book a demo →

www.doppler.com