okta

Adapt to the

Cloud Operating Model

There's a lot of truth to the statement that all companies are technology companies. After all, the core focus of a technology company is to deliver software, whether internally to empower the workforce or externally to serve customers. Technology companies also maintain servers to create, collect, store, and access data—which is now the norm for organizations worldwide, whether public or private, commercial or enterprise.

Yet the software and servers of today are different than they used to be. Software delivery has undergone a transformation with the rise of cloud computing: what were once multi-year waterfall projects to develop and package software for major releases are now continuous, agile streams of regular updates. Similarly, what were once fixed data centers with static servers are now dynamic cloud environments of elastic resources across IaaS providers such as Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure.

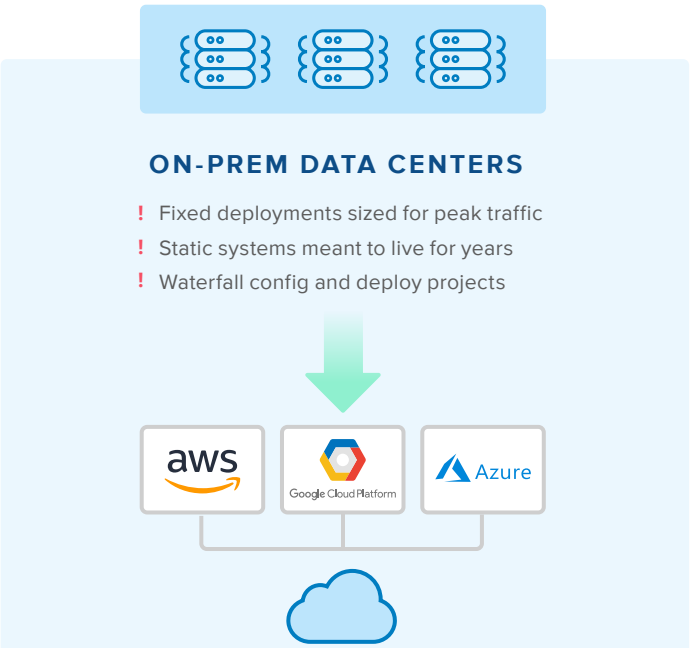
The underlying theme behind software delivery and cloud infrastructure has always been increased speed, as both competitive pressures and growth demand it. Businesses must deliver continuous innovation to their customers, which means developers must deliver continuous innovation back to the business—all in the form of software.

In this virtuous cycle, **velocity at scale** emerges as the primary driver, and DevOps programs are the agent of change as well as the method of achievement. Every company is trying to do more with less, which means streamlining as much digital innovation and process improvement as possible. One way that IT and security teams are embracing this change is by adopting a **Cloud Operating Model**.

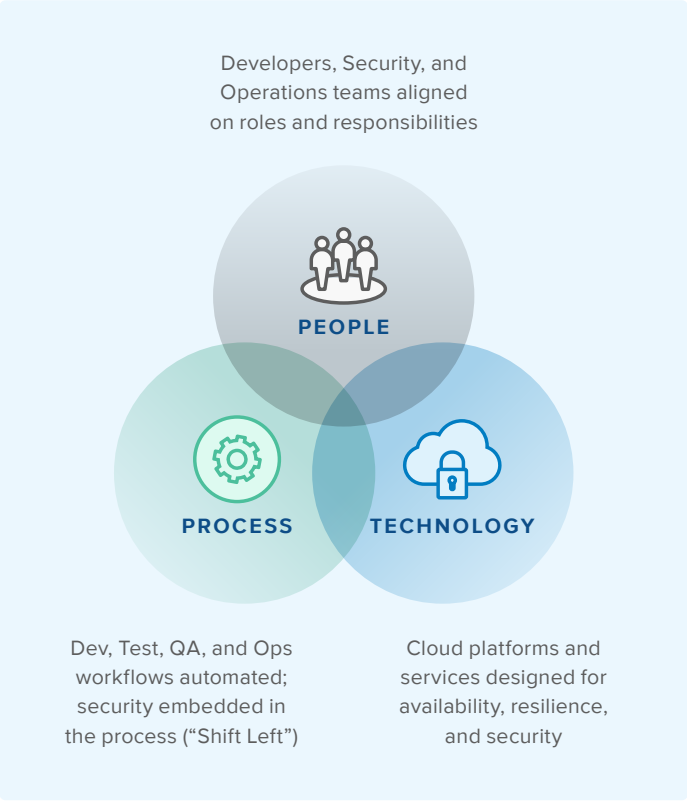
What is the Cloud Operating Model?

This new DevOps-driven paradigm differs significantly from the traditional operating model. Prior to the advent of the cloud, companies would spend large amounts of capital on data center space. These environments were fixed and the servers in them were meant to live for years, with one-time configurations and infrequent updates. And changing these environments required a long, manual, and painful process.

The cloud changed all of this. Companies can now consume a wide array of compute, storage, and networking resources on-demand as pay-as-you-go operating expenses. The environments are dynamic, with elastic resources spinning up and down, adapting to usage patterns in real-time. The resources are only meant to last for weeks, days, hours, or even minutes. Not only that, but these environments are configured via automation tools and workflows so any changes occur almost instantly across the fleet.



The demands for velocity at scale can only be realized by fully embracing the cloud, but with such a significant shift in environmental characteristics and behaviors, the underlying architecture must also be able to adapt. This is the crux of the **Cloud Operating Model**, a strategic framework for cloud adoption—driven by a DevOps culture of automation—that impacts people, process, and technology.



Fostering a culture of automation

Changing DevOps processes has to be preceded by a change in culture. That’s because any agent of change needs buy-in from across the organization to be successful, especially when it spans cross-functional teams. There needs to be alignment on shared goals, shared responsibilities, and shared accountability. Organizations that can achieve this alignment and empower their teams pave the way for this shift in culture, leading to mature DevOps programs.

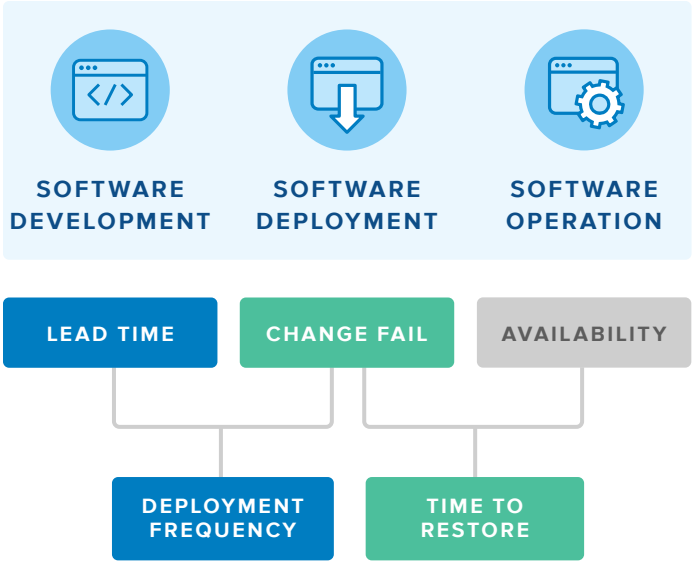
The people, process, and technology of a mature DevOps program are wrapped in automation, with security embedded right from the start. Self-organized teams deploy self-service tools to streamline the delivery and deployment of software, as well as the provisioning and configuration of infrastructure. A key reason cloud adoption requires a shift in the operating model is to remove any barriers from this automation in a secure manner. The traditional operating model is both a blocker to automation and a risk to security, which is bad for the business outcome of velocity.

Measuring DevOps success

Since speed can be quantified, the business leaders who are behind this change will expect results—and naturally, they will expect them quickly. DevOps programs are measured on key metrics related to software development and delivery, while the underlying operations and security functions are evaluated based on how well they support these outcomes.

The DevOps Research & Assessment (DORA) team, now a part of Google, found the following metrics most important for businesses to measure:

- **Deployment frequency:** How often does your organization deploy code to production or release it to end users? *(High performers: between once per day and once per week.)*
- **Lead time for changes:** How long does it take to go from code committed to code successfully running in production? *(High performers: between one day and one week.)*
- **Time to restore service:** How long does it generally take to restore service when a service incident or defect that impacts users occurs? *(High performers: less than one day.)*
- **Change failure rate:** What percentage of changes to production or releases to users result in degraded service and subsequently require remediation? *(High performers: 0–15%.)*



Source: DevOps Research Agency (DORA) State of DevOps Report 2019

The DevOps function doesn't end once software gets out the door; it takes a concerted effort to keep systems and applications healthy. If **throughput** and **stability** are the leading indicators for Dev performance, then **availability**, **resilience**, and **security** are the most important factors on the Ops side. While these attributes have traditionally been considered counter to velocity given their protective nature, organizations with mature DevOps programs understand that they go hand-in-hand. Highly available systems better enable throughput, and resilient systems better enable stability.

Improving system reliability

Popularized by Google, the Site Reliability Engineering (SRE) role complements DevOps programs by taking ownership of **uptime**, widely understood to be the most critical metric of all. Any company that delivers software as a service places their reputation on the line with their Service Level Agreement (SLA); Okta is no exception, with 99.99% uptime SLA.

As an engineering function, SREs are constantly monitoring and improving systems and processes to minimize two key metrics:

- **Mean Time to Acknowledge:** How long does it take to begin working on an issue once an alert has been triggered? *(High performers: less than five minutes.)*
- **Mean Time to Resolve:** How long does it take to resolve an outage and restore service to customers? *(High performers: less than one hour.)*

The dynamic nature of the cloud, with the constant drive for velocity, can make this difficult to manage. Companies with mature DevOps programs tie SRE metrics back to the business, which makes sense because uptime is a competitive advantage. The core engineering work of an SRE team is to build **observability** throughout the entire tech stack, and throughout every automated process. The more proactive they can be with potential bottlenecks, barriers, and breakers, the better.

Where development and operations meet

A fully automated software development, delivery, and deployment program doesn't just happen spontaneously. Most businesses have technical debt to modernize and processes to streamline. DevOps is the unity of software development and infrastructure operations, but each carries their own independent roles and responsibilities. Maturity is defined by how efficiently it all comes together.

Inefficiencies exist when these functions are disjointed, such as when developers write code in isolation and then "throw it over the wall" for the operations team to deploy. This siloed approach can lead to problems when flaws are discovered, or when the two sides have different standards of when a software is considered finished. The famous tagline, "It worked on my machine," originates from this occurrence.

DEVELOPERS

PRODUCT MANAGERS | SOFTWARE ENGINEERS | QUALITY ASSURANCE

Tasks

- ✓ Project management
- ✓ Application development
- ✓ Source code management
- ✓ QA and test
- ✓ Build and package

OPERATIONS

SYSTEMS ADMINISTRATORS | NETWORK ENGINEERS | IT OPERATIONS

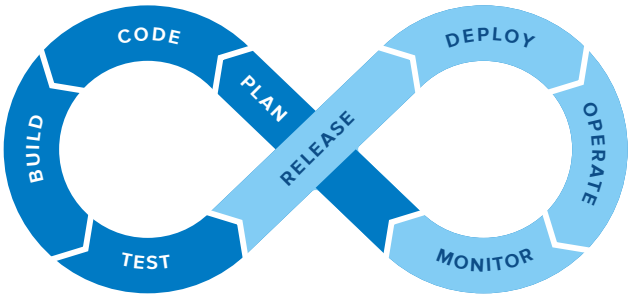
Tasks

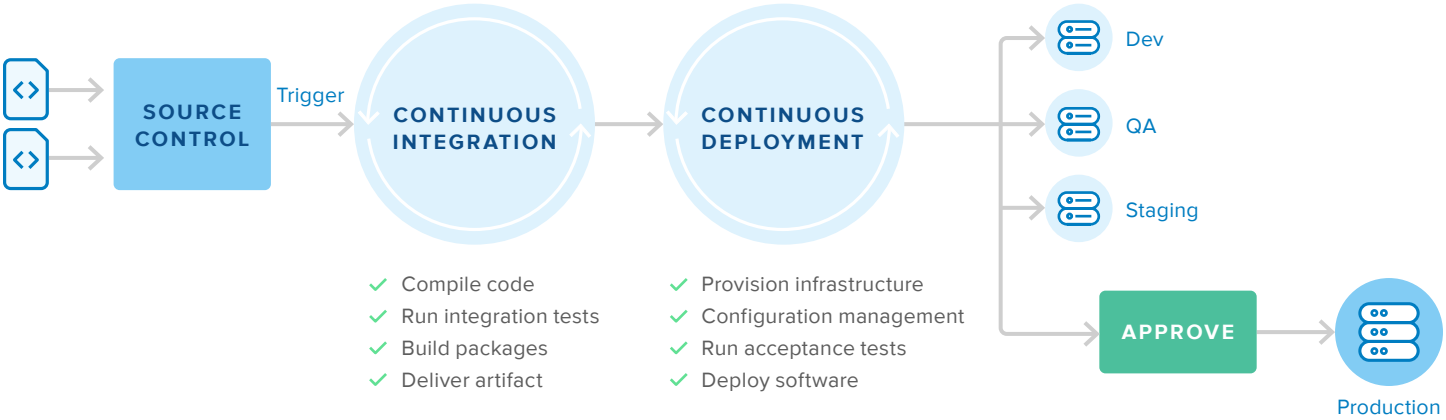
- ✓ Release management
- ✓ Infrastructure provisioning
- ✓ Configuration management
- ✓ Monitoring and alerting
- ✓ Incident response management

The rise of collaboration tools and the advent of virtualization technologies in the cloud have helped bridge this gap. With Virtual Machines and Containers, for example, local runtime environments can closely mirror production environments. In addition, software development and delivery processes can flow through event-driven actions connected to the source control, most commonly Git. It's the various checkpoints along these Continuous Integration and Continuous Deployment (CI/CD) pipelines that enable better collaboration.

While the image below is a significant improvement compared to the legacy approach, there is still much to be desired. Dev and Ops teams may still be disconnected, and the checkpoints may still contain heavyweight, manual processes. The "holy grail" state is to fully automate everything in a continuous cycle. For any business, the ideal is for developers to only be writing code, and for operations to only be optimizing for performance.

As with any ideal state, this view of a continuous cycle of automation and collaboration may seem impractical in reality. But there's nothing wrong with striving for the "holy grail" so long as teams take the right incremental steps to get there and proceed with caution.





Balancing productivity and security

Adopting the Cloud Operating Model as discussed means approaching initiatives and challenges with a balanced mindset—in particular, balancing **productivity** and **security**. There are generally two perspectives with regards to this:

- **Productivity without compromising security** is the dev-centric perspective, taking great care that the automation being built factors in security considerations before deploying live in production.
- **Security without impacting productivity** is the ops-centric perspective, taking great care that the policies and procedures required to meet compliance guidelines don't get in the way of the automation that is built.

These perspectives are really two sides of the same coin—enabling **secure velocity at scale**. When all the functions of DevOps are aligned in this way, the culture of automation really begins to take form, delivering the necessary continuous innovation back to the business for the win.

The critical role of identity

Identity is at the heart of any organization, forming the foundation for a productive and secure workforce, and enabling digital teams to build lasting and scalable customer experiences. In the cloud, identity also covers the full spectrum of people, process, and technology. As such, it plays a key role in the evolution towards the Cloud Operating Model.

One reason why identity is central to a successful Cloud Operating Model is because DevOps programs are in perpetual and recurrent cycles of change.

- **Technology changes:** Dynamic cloud infrastructure environments are made up of ephemeral resources constantly spinning up and down, which change the surface area.
- **Processes change:** Streamlined software delivery pipelines and automated infrastructure provisioning change the workflows.
- **People change:** Teams collaborating to design secure automation at scale regularly change their individual and collective roles and responsibilities.

The change brought upon an organization has a significant impact, and requires careful attention to keep processes on track. With velocity as the business driver, **identity is the key** for organizations to move fast—without breaking things.

Adding security to DevOps programs

In an environment and culture of constant change, new considerations appear that must be addressed early and often. From an infrastructure operations perspective, a changing surface area means controls need to be able to adapt to dynamic resources. But since it's difficult to restrict access to tools, servers, and applications when their locations are always in flux, this poses a challenge for security. That's why security needs to be built into the DevOps process—a practice known as **DevSecOps**.

The annual State of DevOps Report commissioned by Puppet, Splunk, and CircleCI focused on security in their 2019 edition, and the key takeaway was glaring:

> *"For most companies we've spent time with though, integrating security into the software delivery lifecycle is an **unrealized ideal**, and an obstacle to furthering their DevOps evolution."*
>
> — State of DevOps Report 2019

The business outcome of velocity isn't limited to pace. The change across the organization introduces difficult challenges with regards to **speed**, **scale**, and **complexity**. Practitioners who are on the hook for this change may relate to the following statements:

- **Speed:** "We can't keep up with the pace of the business unless we make some compromises."
- **Scale:** "We're operating distributed cloud systems at a crazy scale right now, things are bound to fall through the cracks."
- **Complexity:** "We're still trying to figure out how to just observe all of our cloud resources, let alone protect them."

These statements don't demonstrate indifference, but desperation. Exacerbating this challenge is the fact that any potential exposure only compounds at scale, and a seemingly minor slipup can have devastating consequences. The recent $80 million Capital One settlement that led to 100 million customer data records being leaked originated from a single errant wildcard.

This is why it's critical for organizations to adopt a culture of secure automation as they build out their DevOps programs, bringing together **Infrastructure as Code** with **Policy as Code**.

Bringing security closer to the people

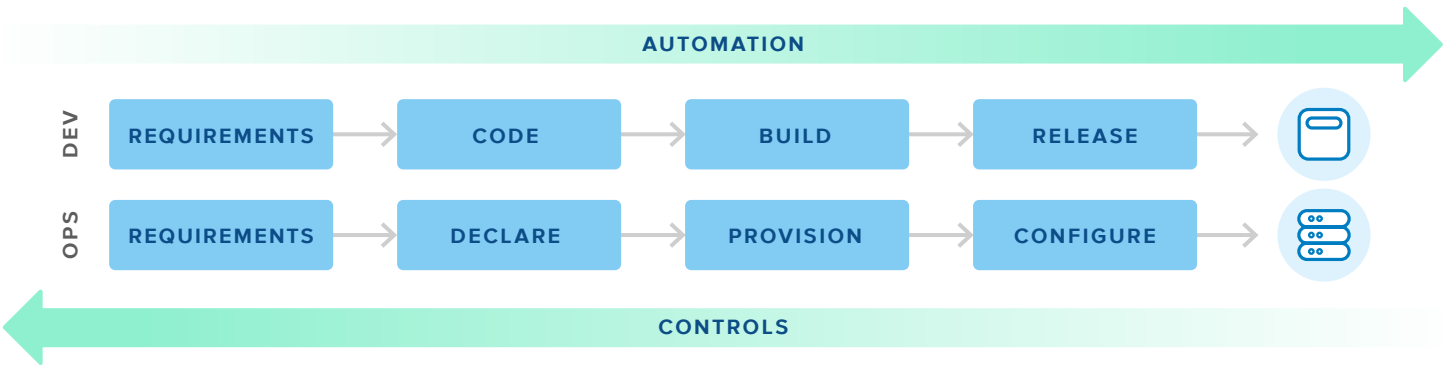
Continuous innovation drives the need for continuous delivery, and continuous delivery drives the need for continuous security—but it must be an enabler, not a blocker. The term **Shift Left** means embedding security controls early and often in the development process, when human decision making is still involved. This applies equally to software developers writing application code as it does operations engineers writing infrastructure code.

Shifting Left doesn't simply mean placing more responsibility on the developers. On the contrary, the intention of DevOps programs is to streamline operations so developers can ship software faster and more effectively. Similarly, DevSecOps is meant to streamline security controls and checkpoints to enable the same.

Factoring identity and access management into DevSecOps

A fundamental aspect of development, operations, and security is identity and access management (IAM). Systems that power software applications consist of numerous tools and infrastructure, with a consistent need for IAM functions:

- **Provisioning and deprovisioning accounts:** Systems have user accounts that need to be managed across their lifecycle.
- **Performing authentication and authorization:** Users need to log into systems to perform tasks, which require strict contextual access controls.
- **Issuing credentials for apps, APIs, and systems:** Contextual access decisions need credentials to match, which range from passwords to keys to certificates.
- **Brokering secure connections to resources:** Credentials are used to establish secure connections with target resources across a range of protocols including HTTPS and SSH.



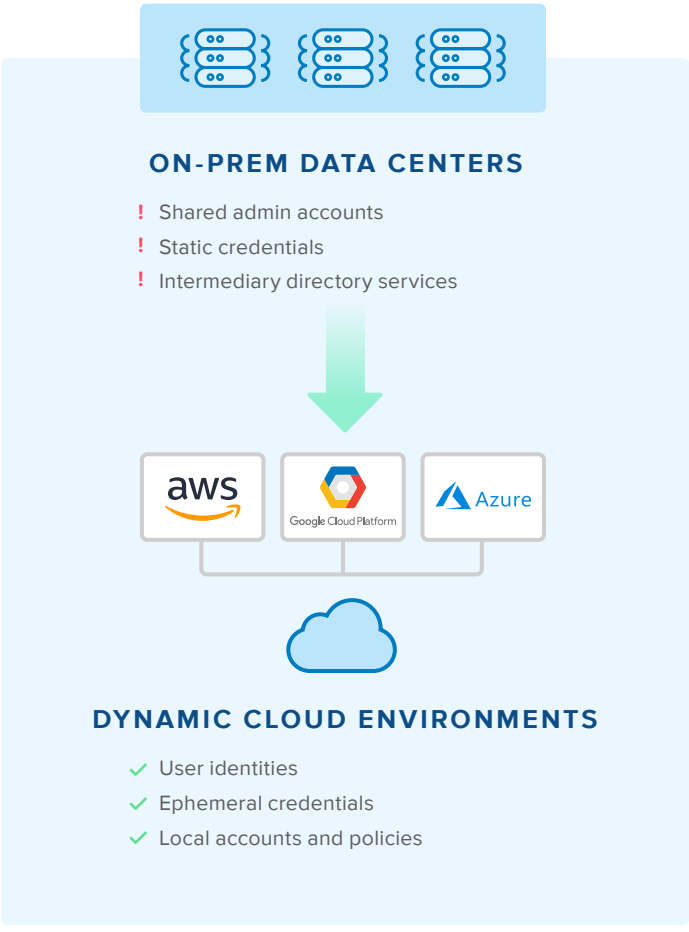
- **Enforcing security policies:** The principle of least privilege dictates that policies should be clear about who can access which systems and when, and what they can do. In order to be effective, policies must be adhered to in practice, which requires a series of enforcement points.
- **Logging and auditing user behavior:** Policy adherence must be proven to achieve compliance, which means there has to be a clear record of what actions users performed throughout the automation processes.

These functions should be factored into automation processes to uncover a number of key questions related to IAM.

Shifting security left requires shifting identity left

As a critical function across the software development and infrastructure provisioning lifecycle, it stands to reason that IAM follows the same Shift Left principles as security. When organizations Shift Identity Left, they make it part of the automation process, not an afterthought.

This is especially important when looking at the attributes of the Cloud Operating Model—elastic multi-cloud deployments, ephemeral resources that spin up and down, and automated delivery and provisioning. With dynamic environments, access management can't be tacked on after the fact like they could when server fleets were fixed and static. IAM must be baked in from the very beginning.



Making identity part of automation puts the right guard rails in place to avoid the dangers of **configuration drift** and **credential sprawl**. The gains are as clear as the outcomes:

- **Identity-first access:** With identity as the foundation, it's easy to specify, enforce, and audit who can access which resources under what conditions, what they can do, and what they did do. This improves security posture and supports compliance initiatives.
- **Dynamic controls:** Modern, automated authorization allows contextual access to be matched with one-time credentials. This enables users to connect to the resources they need without relying on static or shared credentials that can be lost or stolen.
- **Automated lifecycle management:** Admins can ensure accounts are up-to-date with the system of record, and any joiner, mover, or leaver scenarios are applied across dynamic environments in near real-time. Incorporating identity into automation eliminates the problem of cleanup and drift.

Not only does injecting identity as early as possible into automation pipelines minimize the exposure of sensitive accounts and credentials, it enables DevOps teams to completely remove static credentials from code. By replacing them with just-in-time credentials, they can reduce the threat surface and enterprise-wide risk.

This is what **Okta Advanced Server Access** (ASA) was built for. The solution enables just-in-time passwordless authentication to Linux and Windows servers, automates the end-to-end lifecycle of user and group accounts, and enforces role-based access controls and command-level permissions.

How to begin the DevSecOps journey

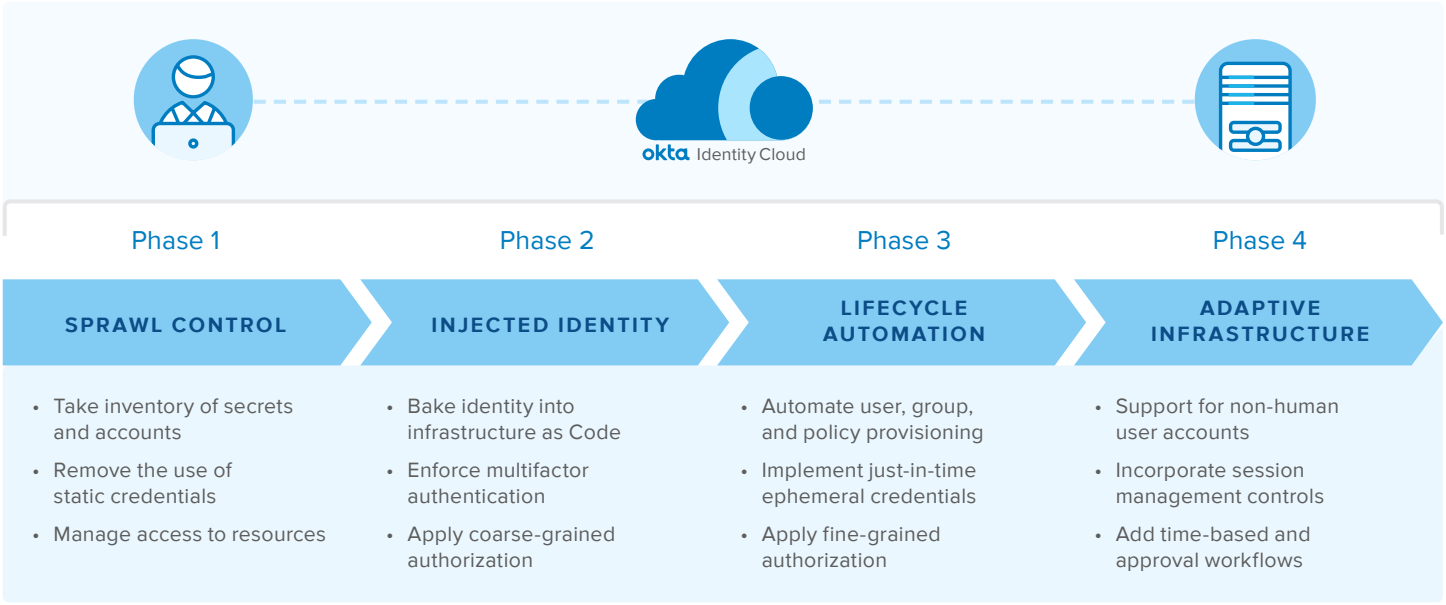
Any organizational change that impacts people, process, and technology needs to be taken in stride. Much like climbers couldn't reach the peak of Mt. Everest before first getting to Base Camp, organizations shouldn't adopt the cloud before laying the foundations for security.

Identity and access shouldn't be an afterthought. Meeting the demands of speed, adhering to the principles of least privilege, and securing velocity at scale, requires a modern, proactive—and automated—approach to IAM:

- **Replace static credentials with just-in-time credentials:** Static credentials pose a significant risk of sprawl. Dynamic systems need dynamic credentials to match.
- **Make all of your security policies identity-centric:** People are dynamic, too—permissions should be tightly tied to roles and attributes.
- **Automate lifecycle management:** With a strong directory service, user accounts and policies can be easily managed across all downstream resources.

Finding your place on the DevSecOps Identity Maturity Curve

With the understanding of how a strong identity foundation can help with the adoption of a Cloud Operating Model, let's look at the journey through the lens of DevSecOps Maturity Identity Curve, and what's needed to progress through each phase.



PHASE 1: SPRAWL CONTROL

Start by taking inventory of all accounts and eliminating the use of static credentials. This will allow you to place more effective access controls between your users and your resources.

PHASE 2: INJECTED IDENTITY

Ensure identity is embedded into your Infrastructure as Code by automating account provisioning. Because these are highly privileged activities, it's best to incorporate additional authorization steps and access controls such as multi-factor authentication.

PHASE 3: LIFECYCLE AUTOMATION

Automate the end-to-end lifecycle of administrative accounts and policies. In practice, this means being able to adapt to any joiner, mover, or leaver scenario in near real-time.

PHASE 4: ADAPTIVE INFRASTRUCTURE

It's perfectly reasonable to stop at Phase 3—but if you really want to get to the peak of Mt. Everest, apply the previous three steps to all your users and service accounts.

A complete solution for a Cloud Operating Model

Identity is critical to a successful DevSecOps program. Having policies that are clearly attributed to specific users and roles, centralized access controls across hybrid and multi-cloud environments, and automated lifecycle management sourced from a system of record are what allow these processes to work at scale:

- Single Sign-On (SSO) and Adaptive Multi-Factor Authentication (MFA) allow streamlined authentication and contextual access controls for server logins.
- Okta Universal Directory (UD) acts as the single source of truth for servers.
- Okta Lifecycle Management (LCM) automates the provision of server accounts from end to end.
- Advanced Server Access unleashes the full benefits of the Okta platform and extends core IAM functionalities to any cloud environment and infrastructure.

Additionally, ASA allows admins to assign Okta users and groups to projects which encompass all servers they require access to. Once users and their devices have been authenticated in Okta, they can be authorized for any projects they belong to through role-based access controls; this is done by minting a short-lived and tightly-scoped client certificate, which is injected with metadata from the user ID and device ID.

But the best part of all this is how UD and LCM enable these workflows to scale across tens of thousands of dynamic servers spanning multiple clouds, all spinning up and down constantly. Teams only need to decide which Okta groups to assign to Okta ASA; from there, they can configure who has access to which servers in ASA, and fully automate account lifecycles.

Identity and access management, like security, must be embedded into the cloud operating model through automation, so it never becomes an afterthought for DevOps and security teams. By bringing the best-of-breed solutions to your dynamic infrastructure, the Okta Identity Cloud does just that.

Get in touch to find out how Okta can help you adopt a Cloud Operating Model, or check out the following resources to learn more about DevOps:

- Advanced Server Access & Your Journey to the Cloud Operating Model (Webinar)
- Adapting to the Cloud Operating Model: Using Okta + HashiCorp to Automate Identity + Infrastructure as Code (Blog)
- What Is DevOps? (Video)


okta