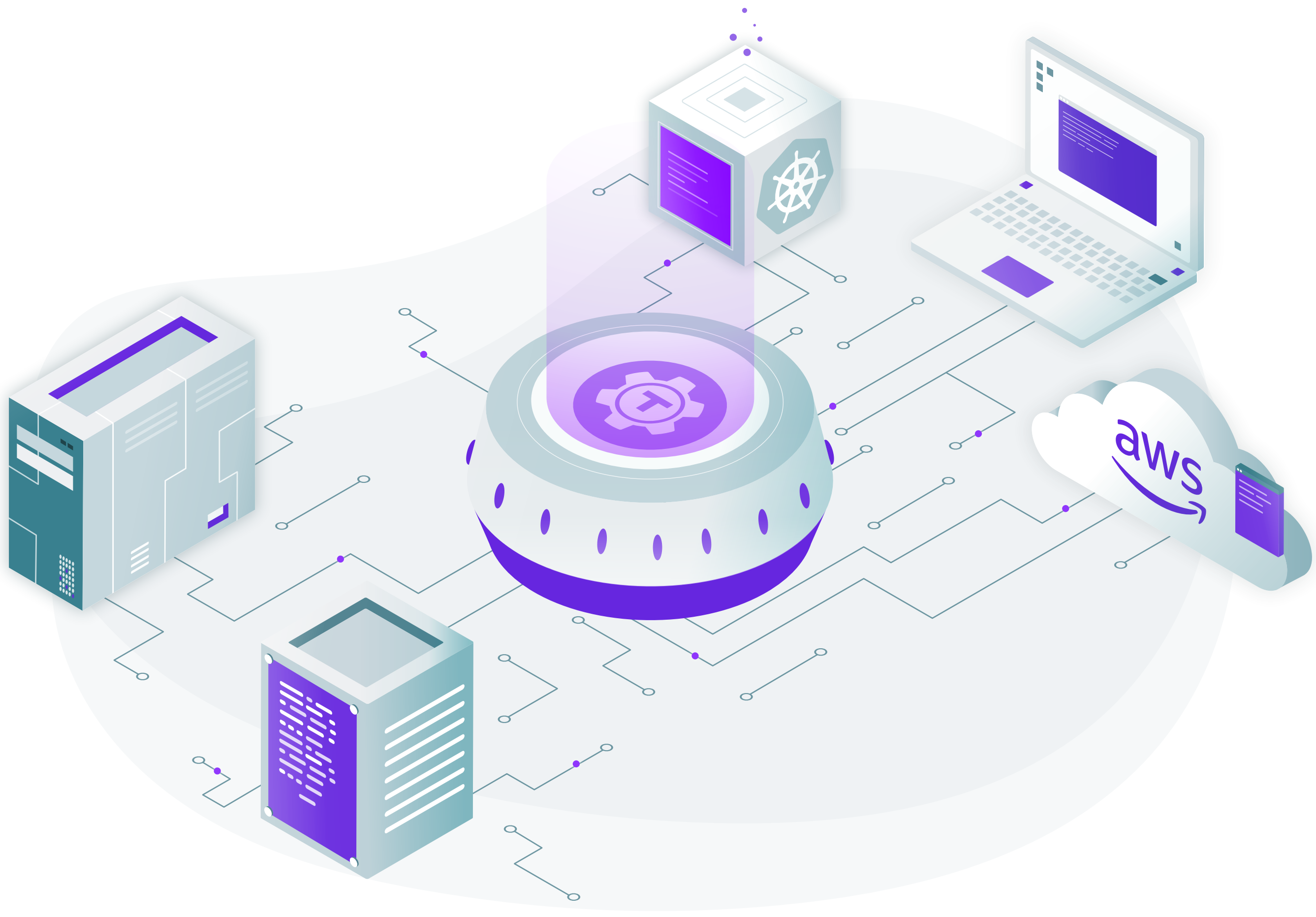
November 2020

Environment-free Computing

Do compute everywhere



Teleport

Environment-free Computing

Do compute anywhere

By Ev Kontsevoy

November, 2020

Ev Kontsevoy is Founder and CEO of Teleport, a leader in environment-free computing. In addition to pursuing his mission of enabling engineers to quickly access any computing resource anywhere on the planet, Ev spends his time hacking, investing, and advising startups.
ev@goteleport.com | **@kontsevoy** on Twitter

I. Executive Summary

Despite advances in software development tooling and the maturing of DevOps, today's cloud applications and infrastructure are excessively complex and increasingly fragile. Unlike desktop and mobile applications, cloud-native software requires 24/7 monitoring and management. This is due largely to the rise of API-driven infrastructure, increased complexity in application architecture, fragmentation of computing environments, and industry reliance on legacy remote management protocols inherited from the client-server era.

Our dependency on legacy client-server access technology has created three main problems: ineffective security, complicated access, and increased operational overhead. Every company today bears the increasing costs of managing many computing resources across multiple computing environments. Fragility within computing environments is the biggest weakness of cloud computing. Their multi-layer architectures make them expensive to access, secure, and manage.

So how do we solve these problems and achieve secure remote access without destroying developer productivity?

Let's briefly consider the history of secure access in computing: a boom in internet growth required the ability to process data at scale, which required an enormous number of computing resources, leading to the necessity of computing environments. These environments are the root problem around secure access - so what if environment-free computing were a possibility? In this paper, we will argue it is, and the solution is here sooner than you might have considered.

"Engineers should have the freedom to compute anywhere, with fewer barriers and inherently more trust."

Engineers should have the freedom to compute anywhere, with fewer barriers and inherently more trust. This ideal future state compels us to envision all computing resources of an organization, such as servers, storage, databases, internal applications, etc. regardless of their location, as a single geographically distributed computer protected by a Unified Access Plane (UAP).

While the engineering community should rally to advance environment-free solutions across all stages of the software lifecycle, this paper argues for the primacy of **simple secure access** to all computing resources. Current approaches to secure access are fundamentally limited and engineers need simplified secure access to servers, applications, databases and other resources across multiple environments and geographies. To achieve this, we must look beyond the zero trust paradigms. We must design for simple, frictionless resource access that provides security teams complete visibility into behavior across all computing resources and all environments.

II. A Brief History of Computing

To better understand how the complexity of computing environments and its impact on security and access came to be, let's briefly take a look at the history of computing in three major stages.

The Classic Stage

The classic stage saw a transition from hardware to software. Computer programs were designed to run on a single computer, which is still the computing model for most of the desktop applications we run locally. This stage is characterized by an inherent dependency between software and hardware. The bigger the task, the more complex software needs to be, and the more powerful the computer must be. Early computers were sold pre-packaged with the software designed to run on a specific machine. Computer manufacturers ruled supreme. Secure access meant physical access to the hardware.

The creation of the operating system reduced the hardware/software dependency by creating useful abstractions between the two. This allowed developers to write applications that could run on any computer with a supported operating system. The advent of the OS also enabled software distribution and gave way to the software industry. The importance of hardware began to wane, and the software lifecycle became defined by three activities: development, access (use) and maintenance. Operating systems added built-in mechanisms for securely accessing computers: being in front of a machine was no longer enough, one had to know the password to login.

The Client-Server Stage

The client-server stage of computing emerged next as individual machines were becoming more powerful and growing in number. The use of computer networks allowed flexible sharing of computing resources between users. A new form of resource sharing was designed where one computer, called a server, ran programs and stored data for multiple users connected over a network. Individual users ran less powerful computers called clients. This model of computing was called client-server architecture.

Multi-user applications grew more complex and started to consume additional computing resources such as databases, message queues, and other middleware. Programmers were generally developing client-server software as they had done before, but the client-server model introduced problems across other parts of the software lifecycle.

"As computing architecture evolved the issue of simple secure access was ignored because co-located engineers and machines meant physical access security was deemed to be ok."

As users enjoyed sharing data and programs, however, no single user was responsible for maintaining the server and its operating system. Individual users could no longer install or upgrade their programs without disrupting other users. Access also quickly became problematic in the client-server era for a few reasons:

1. Access became remote (via the network), so physical security of a server room was no longer enough
2. Users and system administrators required different privileges
3. Several flavors of remote access were needed: operating system access, application access, and middleware access to the computing resources that stitched applications and servers together

These new security and access problems were eventually addressed by several different vendors. Fragmentation of access started to take hold. Operating system vendors added remote access facilities to the operating system. Middleware vendors added remote access to their products such as databases. Network vendors created solutions such as VPNs to create a networked version of physical security, sometimes referred to as perimeter network security.

While these numerous vendors were successful in solving the individual problems, the resulting complexity of maintaining and accessing applications during the client-server era gave rise to a new role called System Administrator. This is a mission-critical role with the responsibility of remotely managing operating systems, applications, and databases. Dependency on Sys Admins subsequently grew quickly.

The Cloud Native Stage

The complexity of applications and systems combined with the explosive growth of the internet led to today's cloud-native stage of computing. As the number of internet users grew, so did the size of data they wanted to process. Scale quickly became the central challenge. Early attempts to scale the client-server model focused on vertical scale (i.e. upgrade to bigger servers). But vertical scale quickly proved impossible to keep pace with a global network of internet users, no matter how powerful the server.

Horizontal scale addressed the problem. Using many relatively inexpensive servers horizontally to run a single piece of software was a colossal shift for the industry and it had major impacts on the complexity of computing. Instead of a handful of servers with a few middleware dependencies, cloud-native applications suddenly required computing environments consisting of large numbers of computing resources. Computing resources include software-defined networks, storage volumes, numerous middleware components, internal dashboards, and so on.

"Computing environments or tech stacks solved one problem, but the increase in complexity especially in the arena of access and security impedes developer productivity."

The computing environment, i.e. a group of computing resources, also sometimes called a tech stack, was born. Today's computing environments are essentially custom cloud computers designed to run a large number of specific applications. This has resulted in complexity at scale, but also an additional set of security and productivity issues due to a reliance on legacy access solutions.

III. The Secure Access Issue in the Cloud Native Era

The custom cloud computer, as characterized by these new and complex environments, does not readily support a standardized approach to secure access. The access is fragmented. This cloud computer consists of thousands of machines and computing resources, each with a unique remote access protocol inherited from the old client-server days. Our dependency on legacy client-server access technology creates three main problems in the cloud-native stage:

1. Ineffective Security

a. **Lack of identity awareness.** Each computing resource (database, machine, internal dashboard) often has its own database of users. Even if a directory service is employed, the directory is often difficult to manage and is available only within a single environment. The directory service is also often disconnected from the single sign-on (SSO) system used by the rest of the organization.

b. **Reliance on shared secrets.** Traditional remote access protocols rely on a static set of (often shared) credentials such as SSH keys or API keys. The theft of a shared secret is a serious security threat. This issue is magnified by the recent shift to remote work, wherein engineers use their own devices to access corporate environments. This access often goes unmanaged even after engineers leave a company. The industry has responded by adopting secrets management solutions that further increase the fragility of computing environments.

2. Complicated Access

a. **Lack of Visibility.** As the number of computing resources and environments grows, there is no centralized visibility into access and behavior across them all.

b. **Intermittent Connectivity.** Cloud applications are becoming more distributed with some microservices running on the edge, behind NAT, with frequently changing IP addresses, without public access, and without reliable Internet connections. How do you SSH into a self-driving truck?

c. **Fragmentation.** Legacy access technologies are hindered not only by network boundaries, but also by organizational boundaries. It's nearly impossible to federate access to shared computing resources among organizations or teams. How can an organization carve out just one portion of their computing resources — MongoDB databases and the servers they run on, for example — to be externally managed by a service provider without giving MSP access to the rest of their environments?

3. Operational Overhead

a. **Endpoint Sprawl.** Each computing resource (database, server, internal dashboard, Kubernetes cluster) has its own access protocol, leading to the sprawl of access points. This increases the operational overhead of managing access and forces organizations to simplify access management by relying on network perimeters.

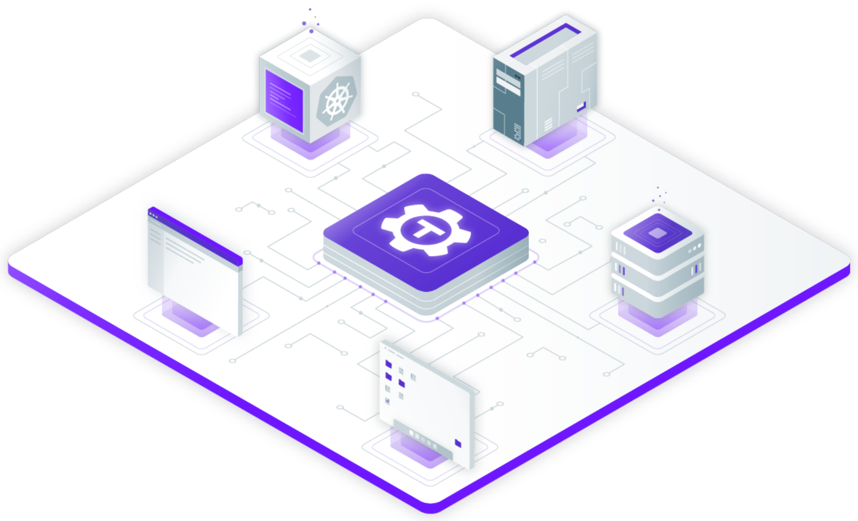
b. **Network Boundaries.** This approach to security requires each computing resource to be on the same network as every client, creating the need to deploy VPN-like solutions. This not only hurts developer productivity but also adds another layer of complexity and increases operational overhead.

Achieving secure remote access is quickly becoming an acute pain point for many technology leaders as the complexity of computing environments grows and the shortage of DevSecOps, DevOps and Security talent intensifies. For many companies, there are no clear answers to questions like, "What is happening across my production environments right now?" or "Who has access to my production environments across all cloud providers?" or more importantly "Who had access to this resource on this date and what did they do?"

"Achieving simple secure remote access for engineers is an acute headache experienced by all companies."

Many leading technology companies are hacking together in-house solutions to provision remote access for engineers, but such a solution **ultimately forces a choice between the productivity interests of developers and the compliance needs of security teams**. This is why most organizations employ different access strategies for engineering teams and the rest of the organization; a choice driving a dangerous increase in backdoor access points to applications for the sake of efficiency.

IV. The Solution: The Unified Access Plane



How do we address the continued increase in the complexity of securing endpoint sprawl while solving the problems around ineffective security and complicated access without destroying developer productivity?

History provides an answer: Consider the move to client-server and cloud-native infrastructure. A new orchestration layer was required. The same is needed to solve the access problem. It requires a Unified Access Plane (UAP) that provides quick secure access to all resources - infrastructure, applications and data - no matter where these resources reside.

With a UAP, the cost of securely accessing more endpoints with more people is essentially zero - this would enable environment free computing.

Unified Access Plane Requirements

Naturally, the UAP is designed around "never trust, always verify" or zero trust principles, but needs to also fulfill the following additional requirements:

1. **Easy to implement and easy to use:** Engineers can discover and instantly access any computing resource on any cloud, data center, or on the edge. There is no need to manage shared secrets or SSH keys, no need to wrestle with VPNs, or to jump between multiple access points. The UAP needs to do its work transparently to the engineer - like air, it needs to be there but almost invisible to the engineer. The UAP needs to be self-updating, maintenance-free, and lightweight.
2. **Implement Industry Best Practice Security and Compliance:** The UAP needs to implement industry best security practices out of the box and brings all computing resources into compliance with security standards such as SOC2, PCI, FedRAMP, and the like. The UAP provides all necessary access controls, supports single sign-on, multi-factor authentication, and role-based access controls.
3. **Provide unified access across all environments to all computing resources:** The UAP would be used to connect all computing resources - machines, microservices with APIs, databases, etc. This would automatically result in trusted connections between resources, eliminating the need for shared secrets such as API keys and making shared secret management an anti-pattern.

The Path to Environment Free Computing

Consider environment-free computing where the UAP would also offer interesting capabilities for applications developed for internal use, such as dashboards and reporting tools. When exposed via a UAP, these applications would automatically become accessible, identity-aware, and equipped with flexible access controls without any additional programming. Access to computing resources needed by internal applications (like servers, databases, or storage volumes) could also become centralized, elastic repositories of resources under a UAP.

"The unified access plane is the solution to solving the headache of providing secure access to all computing resources."

In short, a UAP would be used for human-to-machine access and for machine-to-machine communication.

The concept of a UAP borrows the best ideas from zero trust and applies them universally to all protocols, all computing resources, and all users. Not only would a UAP increase productivity and product development velocity, it would improve security too. Although the concept of a UAP and environment-free computing is new, the implementation of a UAP is something that we have already started at Teleport. We encourage you to review the Teleport Unified Access Plane and join us on the journey to environment free computing.

V. Conclusion

Environment-free computing is indeed a lofty goal, but it is achievable. Ubiquitous simple access to all resources everywhere is the biggest challenge impeding our progress to an environment-free world. Other projects will need to address the complete software lifecycle, but making software simpler to build and enabling it to run anywhere in the world is a vision we can move toward by collaborating on a Universal Access Plane (UAP) that erases network and security boundaries between environments. Join us today on this journey at goTeleport.com.