Zero Trust Access to Private Apps in Microsoft Azure with Zscaler Private Access™

Reference Architecture

Contents

About Zscaler Reference Architectures Guides

The Zscaler™ Reference Architecture series delivers best practices based on real-world deployments. The recommendations in this series were developed by Zscaler's transformation experts from across the company.

Each guide steers you through the architecture process and provides technical deep dives into specific platform functionality and integrations.

The Zscaler Reference Architecture series is designed to be modular. Each guide shows you how to configure a different aspect of the platform. You can use only the guides that you need to meet your specific policy goals.

Who Is This Guide For?

The Overview portion of this guide is suitable for all audiences. It provides a brief refresher on the platform features and integrations being covered. A summary of the design follows, along with a consolidated summary of recommendations.

The rest of the document is written with a technical reader in mind, covering detailed information on the recommendations and the architecture process. For configuration steps, we provide links to the appropriate Zscaler Help site articles or configuration steps on integration partner sites.

A Note for Federal Cloud Customers

This series assumes you are a Zscaler public cloud customer. If you are a Federal Cloud user, please check with your Zscaler account team on feature availability and configuration requirements.

Conventions Used in This Guide

The product name ZIA Service Edge is used as a reference to the following Zscaler products: ZIA Public Service Edge, ZIA Private Service Edge, and ZIA Virtual Service Edge. Any reference to ZIA Service Edge means that the features and functions being discussed are applicable to all three products. Similarly, ZPA Service Edge is used to represent ZPA Public Service Edge and ZPA Private Service Edge where the discussion applies to both products.

Notes call out important information that you need to complete your design and implementation.

Warnings indicate that a configuration could be risky. Read the warnings carefully and exercise caution before making your configuration changes.

Finding Out More

You can find our guides on the **Zscaler website** (**https://www.zscaler.com/resources/reference-architectures**).

You can join our user and partner community and get answers to your questions in the **Zenith Community** (**https://community.zscaler.com**).

Terms and Acronyms Used in This Guide

Acronym	Definition
AD	Active Directory
AMI	Amazon Machine Image
CA	Certificate Authority
DC	Data Center
FQDN	Fully Qualified Domain Name
FTP	File Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IdP	Identity Provider
LSS	Log Streaming Service
NAT	Network Address Translation
NSS	Nanolog Streaming Service
PKI	Public Key Infrastructure
RDP	Remote Desktop Protocol
SAML	Security Assertion Markup Language
SCIM	System for Cross-Domain Identity Management
SP	Service Provider
SSH	Secure Shell
SSL	Secure Socket Layer (superseded by TLS)
TLS	Transport Layer Security
VM	Virtual Machine
VNet	Virtual Network
VPN	Virtual Private Network
ZCA	Zscaler Central Authority
ZDX	Zscaler Digital Experience
ZIA	Zscaler Internet Access
ZPA	Zscaler Private Access
ZTA	Zero Trust Architecture
ZTE	Zero Trust Exchange

Icons Used in This Guide

The following icons are used in the diagrams contained in this guide.

Zscaler Zero Trust Exchange 

ZIA or ZPA Service Edge 

Zscaler App Connector 

Zscaler Cloud Connector 

Azure Load Balancer 

Azure Application Gateway 

Azure Virtual Machine 

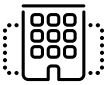
Azure Application or Workload 

Azure VPN Gateway 

Generic Application or Workload 

Azure Autoscale Group 

Private Data Center Location 

Headquarters Office Location 

Branch Office Location 

Factory Location 

Authorized Use 

Bad Actor 

Data Tunnel 

Introduction

Zscaler Private Access (ZPA™) is a part of the Zscaler Zero Trust Exchange™ platform. Based on a zero trust framework, ZPA provides secure access to private applications—including remote access to internal applications running on Microsoft Azure. With ZPA, applications are never exposed to the internet, making them completely invisible to unauthorized users.

The ZPA service enables users to connect to applications via outbound-only connectivity, rather than extending the network to the remote user. In fact, users are never placed on the network. ZPA provides a software-defined perimeter for Azure that supports any device and any internal application.

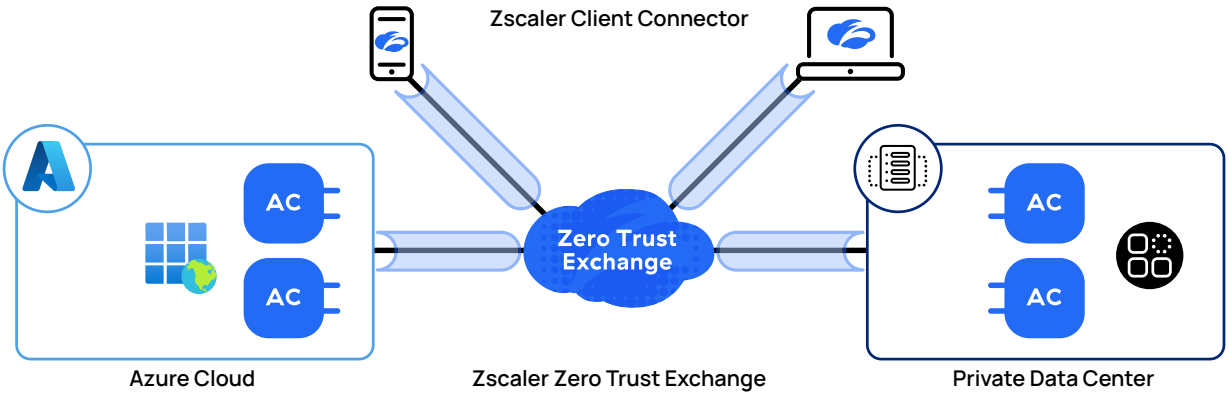


Figure 1. Overview of ZPA deployed in Azure

The previous image illustrates the basic components of the ZPA infrastructure:

1. ZPA Public Service Edge – A Zscaler cloud element that brokers connections between components of the system, with connections to both user agents and the Azure cloud.
2. Zscaler Client Connector – A lightweight app that can provide service to ZPA, Zscaler Internet Access™ (ZIA™), and Zscaler Digital Experience™ (ZDX™).
3. Zscaler App Connector – A lightweight virtual instance that delivers authorized user traffic to applications.
4. Azure ExpressRoute or site-to-site VPN (optional) – As you transition to the cloud, you might have applications in your on-premises data center that are also in Azure. You might also continue running applications or databases in your data center that are accessed via Azure front-end applications. Azure ExpressRoute is used to synchronize those services via a leased line. You can also set up site-to-site VPNs between your organization and Azure.

Combining ZPA with Azure allows you to smoothly transition applications from your on-premises data center to an Azure Virtual Network (VNet). In this guide, we cover the best practices for deploying in the Azure cloud and provide the information you need to be successful in implementing zero trust connectivity.

New to ZPA and Zero Trust?

If this is your first time reading about ZPA, we encourage you to watch this quick video on the benefits of ZPA over traditional VPN solutions at Zscaler Private Access | The 3 Minute Overview (https://youtu.be/1KLbE243dLY).

You can find additional information, case studies, demo videos, and a free test drive of ZPA at Zscaler Private Access (https://www.zscaler.com/products/zscaler-private-access).

Interested in learning more about zero trust? See our zero trust microsite at It Starts With Zero (https://www.zscaler.com/it-starts-with-zero).

Want to delve further into the zero trust architecture? We recommend the National Institute of Standards and Technology paper on Zero Trust Architecture (https://www.nist.gov/publications/zero-trust-architecture).

Solution Overview

The ZPA service provides remote access to applications based on a zero trust architecture (ZTA) model. One of the key points in any zero trust framework is the move away from providing access to a network and instead providing access to approved applications. In a zero trust framework, access is provided to specific applications based on policy. Users receive specific access based on a mix of attributes, including authentication response and device posture.

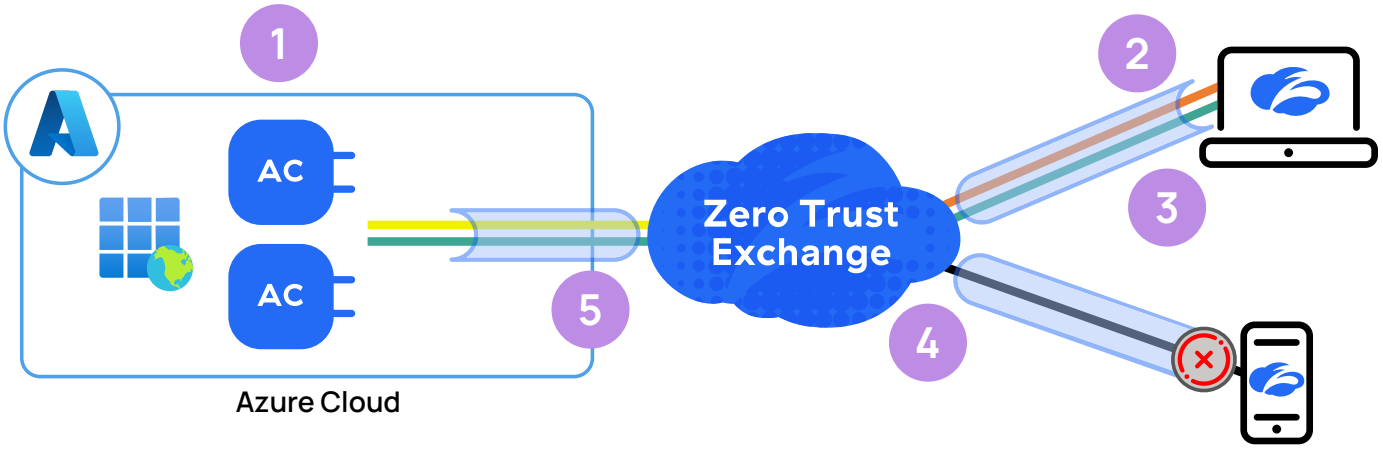


Figure 2. Application access with ZPA

In the previous diagram, one of the users has two devices: a company-issued laptop with Zscaler Client Connector, and a phone that does not have Zscaler Client Connector installed. When the user on a company-issued laptop attempts to connect to an internal application hosted in Azure, such as a company's travel application, the traffic traverses ZPA as follows:

1. The App Connectors are deployed in front of your applications in the Azure cloud. Each connector forms a data-plane microtunnel to the nearest ZPA Service Edge. The applications are only accessible by users via the App Connector(s). The ZPA Service Edge communicates with the App Connectors to validate that they can reach the applications in their local data center or VNet (yellow path).
2. The user's laptop running Zscaler Client Connector establishes its own data-plane microtunnel to the nearest ZPA Service Edge (orange path). The user's phone is also using Zscaler Client Connector and establishes its own microtunnel for communication (black path). This does not have to be the same Service Edge that the App Connector is attached to.
3. The client requests access to an application. The request is intercepted and inspected by Zscaler Client Connector. If the application is marked as internal, it is forwarded across the microtunnel to the ZPA Service Edge. The service edge decides based on policy to allow or deny access to the application. If the application is not an internal app, the traffic resolves normally outside of the ZPA service. This can include tunneling the traffic to a ZIA Public Service Edge.
4. The ZPA Service Edge receives the request and makes a policy-based decision. If the user is allowed to "know" the application exists, then the app resolves and the connection begins. If the user is blocked by policy, the application does not resolve; in effect, it does not exist for that user. No matter the outcome, all policy requests are logged by the Zscaler system. In this case, the user is allowed to access the application on a company-issued laptop running Zscaler Client Connector, but is not allowed to do so from a mobile phone running Zscaler Client Connector. The same user has different outcomes based on policy.
5. The laptop case results in the ZPA Service Edge signaling both Zscaler Client Connector and the appropriate App Connector to begin building a microtunnel (green path). This microtunnel exists only between the user's laptop and the application. Each application has its own tunnel, keeping the traffic isolated from other streams, and is not shared with other users of the same application.

The user who connected to the development server via the company-issued laptop cannot connect using their phone. The phone is not approved as a development environment, and the application's existence cannot be resolved. This makes the application invisible to the user on that device. The application doesn't have a public IP, so there is no way for the user to browse to the application directly.

Even if an internal user knows an app exists, the user cannot necessarily reach it. With traditional VPNs, users are placed on the network, but with ZPA, applications respond to requests with a synthetic IP in a microtunnel specific to that application. There is no lateral movement on the user's part because the user isn't on the network. No access is granted to other applications unless authorized via ZPA.

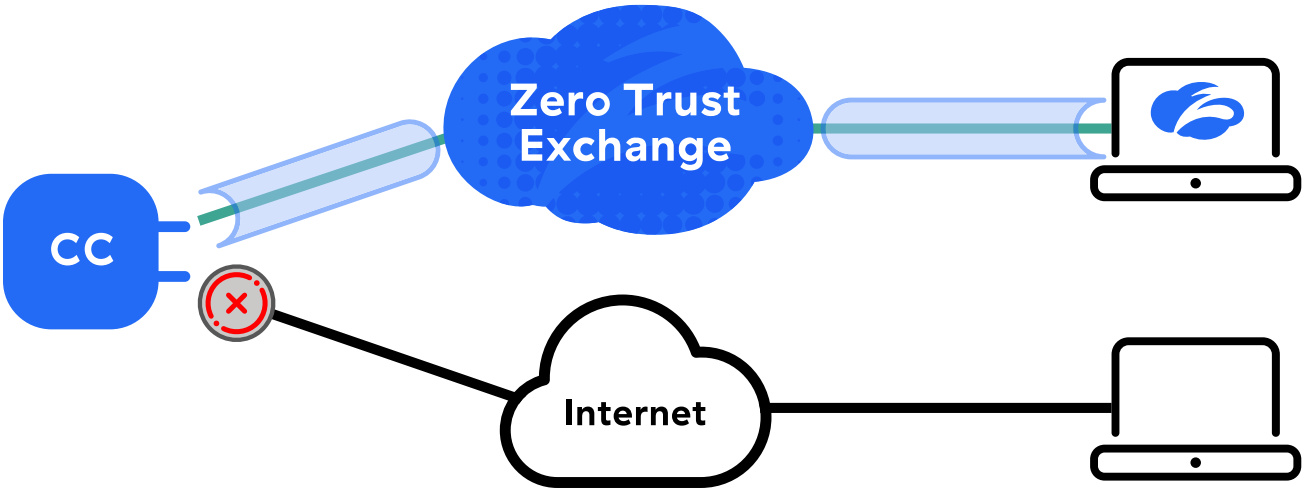


Figure 3. App Connector and Zscaler Client Connector each create outbound-only data-plane tunnels

Because ZPA uses outbound-only connections, the Zscaler App Connectors never accept connections initiated from the internet. These virtual machines instead reach out to the ZPA Service Edge in the Zscaler cloud and establish a TLS tunnel. The control-plane tunnel is used to signal the App Connector when an authenticated user is authorized to access an application. A data-plane tunnel is then established, also via TLS, and a microtunnel is created within the data-plane tunnel, providing access to only that application for only that authorized user.

ZPA also protects your private applications by allowing you to make them invisible to the internet. Since these applications only run privately, you can remove the records from the public side and keep those DNS entries completely internal. This effectively removes them as an attack surface by denying the ability to even resolve their existence.

Zscaler allows you to base user access decisions on multiple factors, including location, device profile, multi-factor authentication, group enrollment, and more. By using multiple context criteria, you can build a fine-grained policy that can change for the same user when that user's access conditions change.

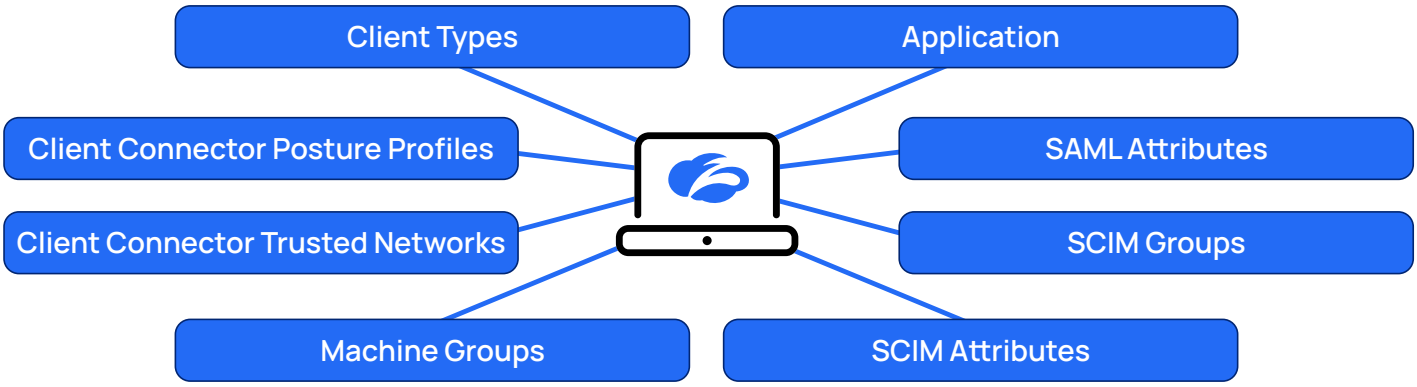


Figure 4. ZPA policy inputs

Differentiating policy on more than a user's role in the organization can help mitigate data leakage or infection. For example, if a trusted user working on an organization-supplied laptop attempts to access financial data from a corporate office, you can provide the normal and appropriate level of access to the application. However, if the same user is at a local coffee shop on a phone that is not a controlled asset, you might want to restrict the user's views or deny access altogether.

Deploying ZPA in Azure

The Azure model makes use of the concept of a virtual network, or VNet, in which applications, databases, and virtual network components such as routers and NAT gateways are all contained within the VNet. Let's look at a simplified model showing only a single Availability Zone. In normal operation, you'll want to deploy your App Connectors across two or more Availability Zones for redundancy.

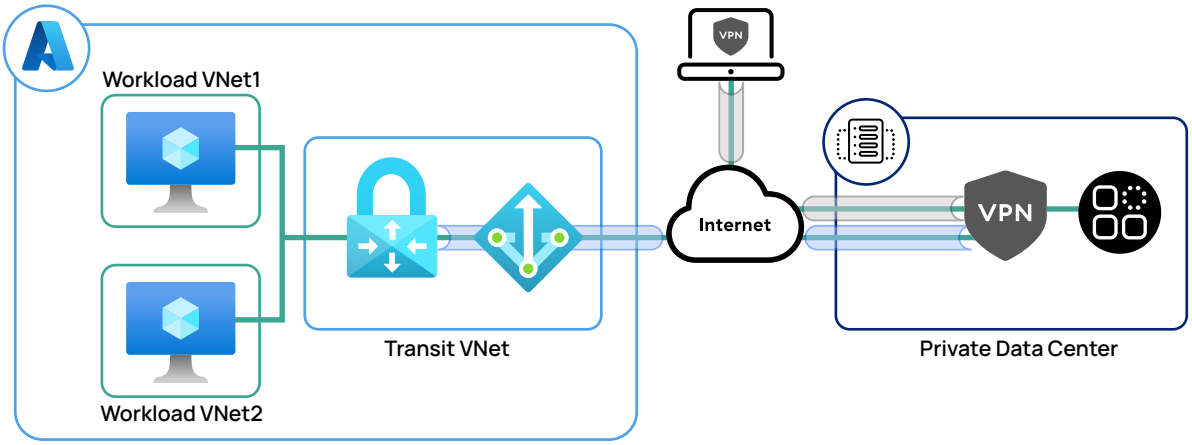


Figure 5. Azure access via legacy VPN

Prior to ZPA, users would likely have taken one of a few routes to access Azure instances. In most deployments, a VPN has been used to provide access to internal applications, which are not exposed to general internet access. Based on this configuration, you could allow users in the corporate office to have direct access to applications while providing those away from the office with a VPN connection. These methods all introduce latency into applications and expose more of the network than is required.

The ZPA model shifts away from backhauling traffic and using a VPN that puts users on the network. The primary changes are the removal of your VPN gateways, as they are no longer needed for access. Your users' traffic no longer takes a trombone-style path through your data center, only to be sent back out to your private cloud.

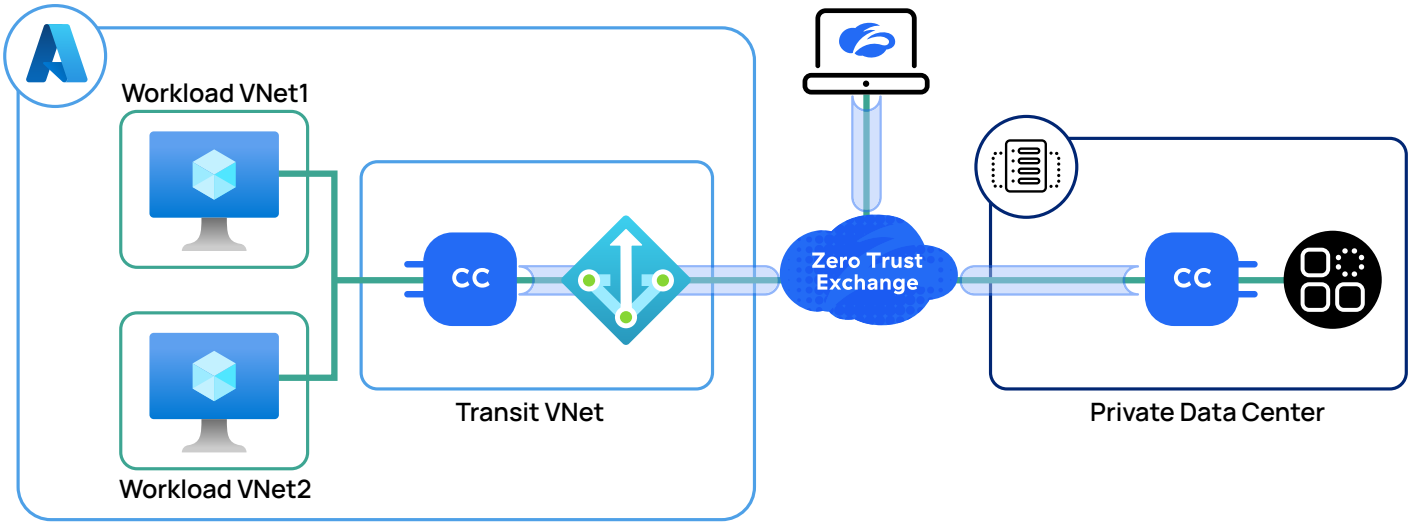


Figure 6. Azure access with ZPA

Instead, you'll deploy groups of two or more Zscaler App Connectors to each of your VNets with internet access. This can be the same VNet that contains your applications, or to a gateway VNet serving private VNets without direct internet access. These are built into logical groups, and access policies are defined for users attempting to reach applications. Zscaler recommends deploying App Connectors in groups of two or more for redundancy, ideally across Availability Zones in a region.

It's important to note that App Connectors are meant to be deployed for the long term. Establishing the number of App Connectors you need to handle your everyday load allows you to deploy them on reserved instances and lower costs.

Your users will run Zscaler Client Connector, which replaces a traditional VPN client. Logging into Zscaler Client Connector provides DNS resolution and access for your internal-only applications.

The Zscaler Client Connector agent also acts as a client for Zscaler Internet Access (ZIA) and Zscaler Digital Experience (ZDX) services. To learn more, see:

- About the ZIA Cloud Architecture (https://help.zscaler.com/zia/about-zscaler-cloud-architecture)
- What is Zscaler Digital Experience? (https://help.zscaler.com/zdx/what-is-zscaler-digital-experience)
- What is Zscaler Client Connector? (https://help.zscaler.com/z-app/what-zscaler-app)

Azure and Hybrid Deployments

Hybrid ZPA deployments are fully supported and increasingly common as applications are moved to public cloud providers. Unlike your old VPN connection, users are not aware of whether they are accessing Azure or the corporate data center. ZPA simply steers the client to the nearest App Connector that can provide application access.

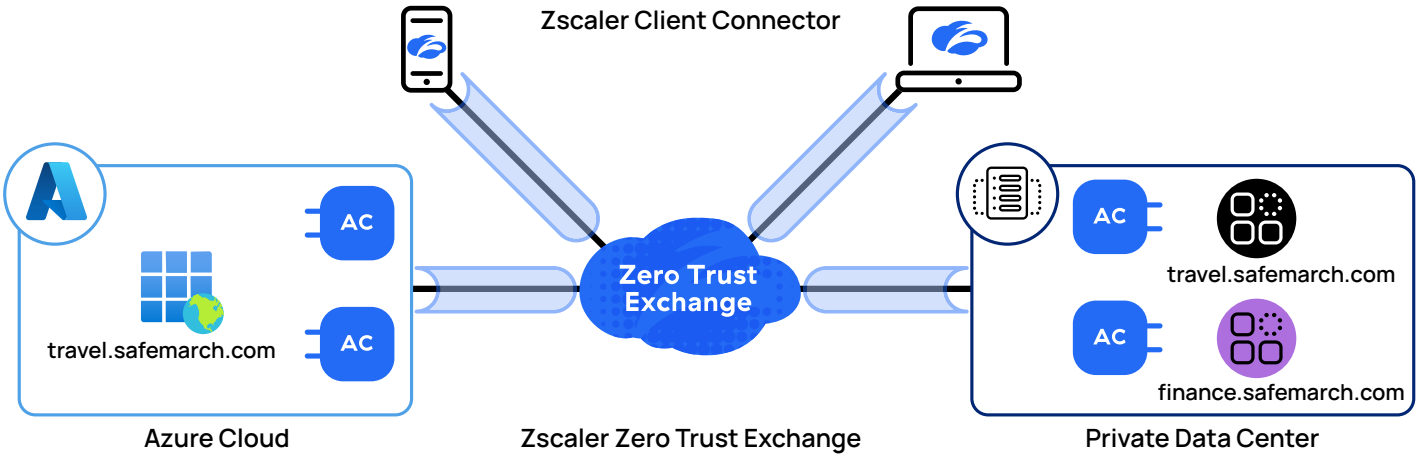


Figure 7. Hybrid access allows apps to exist in multiple places or in your data center only

Adding ZPA to Your Azure Deployment

The process for connecting ZPA and Azure requires deploying virtual machines into your Azure VNets provisioned with access to your applications and the ZPA Service Edges. The App Connectors reach out to the Zscaler cloud to establish a connection and receive configuration. After they are updated, the App Connectors are available to service your user traffic to applications.

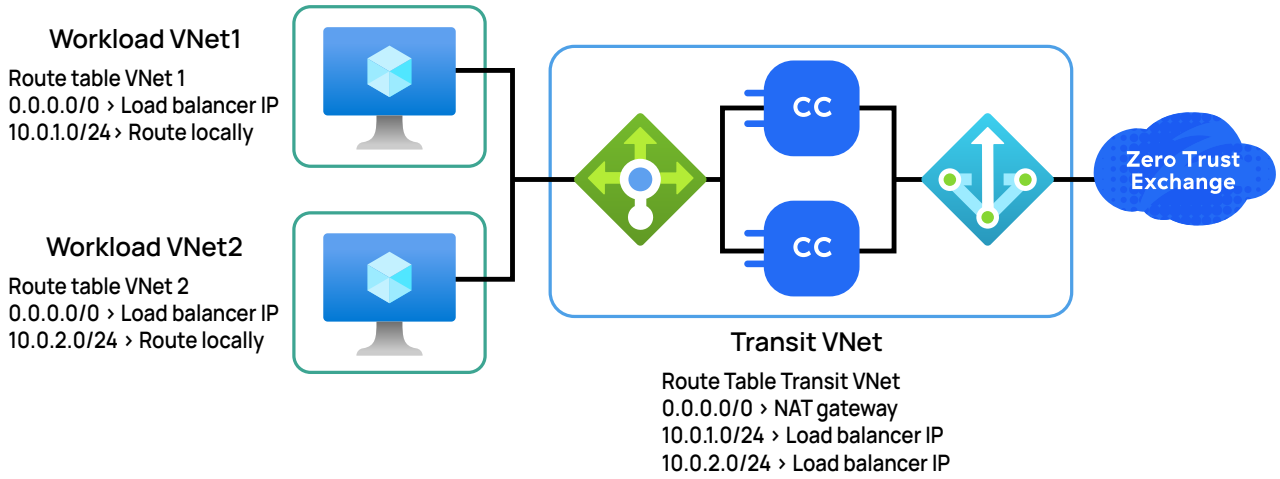


Figure 8. Add Zscaler App Connectors to your VPC

App Connectors

Zscaler publishes the App Connector software in a variety of formats for various platforms. On Azure, Zscaler offers an App Connector in the Marketplace or an RPM for manual installation, both of which are updated regularly. The App Connectors run on the Linux operating system (currently on CentOS 7.2). The Amazon Machine Image (AMI) disables all unnecessary listening services to further reduce your attack surface. This guide focuses on App Connector deployment via the AMI.

The App Connectors are initially provisioned in the ZPA Admin Portal as part of the overall ZPA setup, which includes:

- IdP integration – Zscaler leverages the Security Assertion Markup Language (SAML), relying on your SAML-based Identity Provider (IdP) to perform user authentication. This eliminates the need to manage user accounts directly in the ZPA service.
- Provisioning keys – You can configure per-VNet keys that tie your App Connectors back to your Zscaler tenant. This is how the ZPA Service Edge understands which ZPA tenant the App Connectors supports and what traffic it services.
- App Connector groups – You should logically group App Connectors by location. These groups are used to help define the geolocation of the connectors and the apps the connectors can serve, and they are used by Zscaler to manage regular upgrades.
- Application discovery – ZPA initially helps you identify all the applications you currently have running in Azure by observing user traffic. The system reports on what applications it serves and which users are accessing them.
- Policy definition – After you've discovered applications, you can build granular policies to control access and usage. You can determine what applications are accessible by which users and under what conditions access is acceptable.

On the Azure side, we recommend using Azure templates to consistently spin up App Connectors. These include:

- The approved Azure Virtual Machine instance that can support the App Connector AMI.
- The provisioning key for the App Connectors to contact Zscaler.
- The configuration of Azure Autoscale groups to ensure consistent capacity to the App Connectors.

App Connector Connectivity

Deployment on the Azure side requires minimal configuration changes to your VNET. As the App Connector is the gateway between the user tunnel and the application, you want to ensure that all App Connectors can reach all applications they need to serve.

The App Connectors themselves also need to be able to reach any ZPA Service Edge anywhere in the world. Because you can't control where users might go or what routes their access might take to reach your applications, App Connectors potentially need connectivity to all ZPA Service Edge IP addresses.

The Zscaler best practice is to run the App Connectors behind an Azure NAT gateway. To learn more about Zscaler IP ranges, see **Zscaler Config** (**https://config.zscaler.com/zscaler.net/cenr**).

Updating App Connector Software

The Zscaler App Connector and the host VM both require regular software updates that should be built into your operating procedures. The App Connector runs the Zscaler OS in the virtual machine. Software updates and OS updates are provided by Zscaler via automatic upgrades. When an App Connector is deployed, the software is automatically upgraded to the latest version. The App Connector checks for new software daily and upgrades itself automatically at midnight local time, based on the deployed cloud region.

This automatic check and update means it is critical that your App Connector locations are accurate. An inaccurate location can lead to upgrades occurring in the middle of your day. Always specify exactly where the App Connector is located when deploying the virtual machine.

Zscaler recommends that App Connector appliances be deployed as redundant, high-availability appliances. Specific to software upgrades performed by Zscaler, this ensures that you will incur no downtime. After an appliance is rebooted to accept a new update, Azure Load Balancer automatically moves traffic over to the redundant, active appliance.

There are two components to software updates:

- App Connector updates – Zscaler performs ongoing updates that run on a scheduled basis at a time you select. During a four-hour maintenance window, Zscaler upgrades your App Connectors using a rolling process to minimize user impact.
- Host OS updates – Updating the host OS is your organization's responsibility. The OS can be upgraded using the standard YUM package manager, or by destroying the App Connector VMs and launching new instances.

Because you have control of the update schedule, it's important to group all your App Connectors within a single geographic location into a single group. You can then schedule that group to be upgraded during low-use time periods in that geographic region.

The choice to patch or destroy and rebuild apps will be an organization-specific decision. Teams with robust cloud patching strategies can add these instances to their existing process. Other organizations can destroy and rebuild the connectors from the updated Zscaler image.

If your organization plans to use to use the destroy and rebuild model, you might want to leverage Autoscale groups. This allows you to destroy instances and have them automatically rebuilt to the Autoscale groups' minimum number of instances. See **Leveraging Azure Autoscale for Redundancy** for more information.

Maintaining your instances follows the Azure shared responsibility model. Although cloud IaaS providers such as Azure are responsible for ensuring the security and availability of their infrastructure, organizations are ultimately still responsible for the security of their workloads, applications, and data. To learn more about the shared responsibility model, see **Shared responsibility in the cloud** (**https://docs.microsoft.com/en-us/azure/security/fundamentals/shared-responsibility**).

Defining Policy

Zscaler follows a three-phased approach to defining policy.

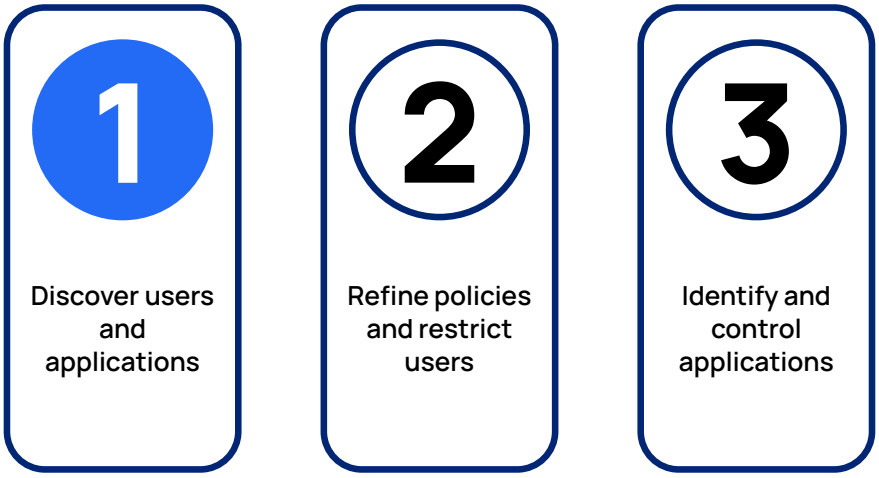


Figure 9. Policy definition framework

Phase 1 – Discover Users and Applications

When you initially deploy ZPA, Zscaler recommends starting your deployment with a limited, controlled group of pilot users and a discovery policy. This policy does no initial enforcement, but simply watches what applications are being accessed by your user community and provides a mapping.

Onboarding users should be done in controlled locations or groups so that you are not initially overwhelmed by application discovery. As your users go about their work, you can develop a map defining which groups need access to which applications.

Phase 2 – Refine Policies and Restrict Users

At the end of Phase 1, you'll know where your users are going and which applications they are accessing. In Phase 2, you'll start locking down known services with a focus on critical applications.

For example, your engineers probably don't need access to your company financials or the accountants to your development platforms. Generally, these are well-known applications with limited numbers of known users.

Phase 3 – Identify and Control Applications

At this point, you'll be discovering fewer and fewer applications, and your users will have settled into the routine of using ZPA. The steps in this phase are refinements of policy.

- Lock down additional services – There will be applications that your employees need to access, either by group or as an organization, but you'll want access to be limited. For example, you might want to make sure contractors or third-party groups cannot access the application. Or perhaps you have regional or country-specific tools that should only be accessible in those regions.
- Decide what to do with shadow IT apps – You'll likely discover applications and services that you were not aware of. These situations often require conversations with groups about what the apps are, why they exist, and who specifically needs access to them.
- Remove or restrict the discovery process – At this stage, most (or all) of your users should be on board and you won't need to continue discovering applications in the same way. You can remove or disable the discovery policy altogether. Or you might choose to retain the policy only for your employees, so that you can continue to discover new application deployments even if you aren't informed about them.

Next Steps

In the next sections, we dive into more detail on each of these topics, providing Zscaler best practices and recommendations for deployment. Each section contains links to the appropriate Zscaler Help documents. These documents show you how to configure the Zscaler service with interface walkthroughs and short overview videos.

Deployments in Azure

Zscaler App Connectors are virtual machines that run in multiple clouds. Deploying Zscaler App Connectors in Azure is like deploying other application instances. An existing Azure Marketplace image is available and is updated regularly by Zscaler. The following section covers core concepts about App Connectors to help you understand and plan your deployment. At the end of this section, you should understand:

- Supported Azure image types
- App Connector connectivity requirements
- App Connector security
- Deploying App Connectors with Azure launch templates and Azure Autoscale

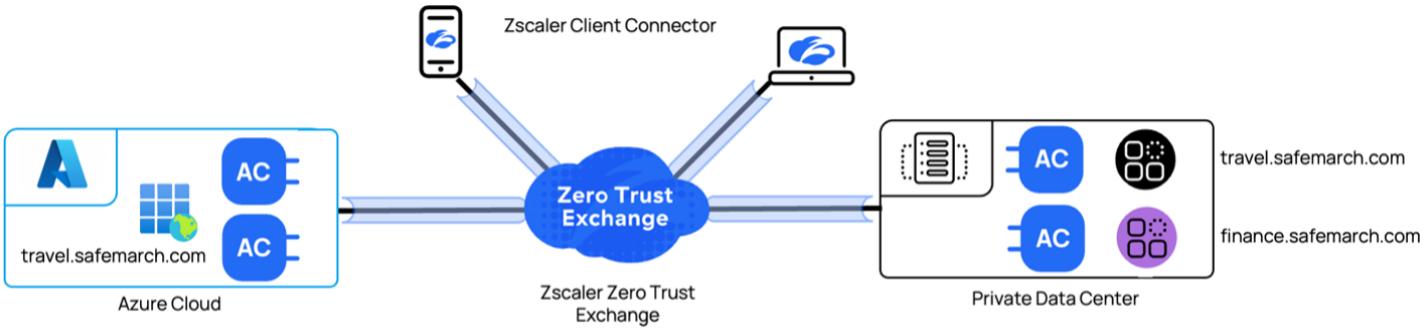


Figure 10. Modified transit VNet with App Connectors

Supported Azure Compute Types for App Connectors

The Zscaler App Connector is available as an Azure Marketplace image. The requirements for the App Connector image, as of this writing, are as follows:

- Memory: 4 GB RAM
- CPU: 4 CPU cores (Xeon E5 class) for virtual machines (VMs) with hyperthreading
- Disk Space: 8 GB (thin provisioned) for all deployment types
- Network Card: 1 NIC

With this configuration, you should see 500 MB of throughput per App Connector. Keep in mind that the double encryption discussed later in this guide impacts App Connector throughput, and you need to account for this difference when sizing your connectors.

For current detailed specifications and sizing requirements, see App Connector Specifications and Sizing Requirements (https://help.zscaler.com/zpa/connector-deployment-prerequisites#SpecsAndSizing).

Azure offers many types of virtual machine sizes, but not all of these sizes are suitable for production use. Zscaler's best practices for App Connector deployment are as follows:

- For standard deployments, Zscaler recommends using Standard_F4s_v2 or Standard_D4s_v3.
- Azure VMs older than V3 require 2 CPU cores, while VMs V3 and higher require 4 CPU cores due to hyperthreading.
- For Zscaler Digital Experience (ZDX) deployments, Zscaler recommends App Connectors to have 4 CPU cores.

Remember that the Zscaler best practice is to deploy App Connectors in groups of two or more across Availability Zones.

To learn more about the differences and details of the Azure VMs, see Virtual Machines (https://azure.microsoft.com/en-us/services/virtual-machines/).

To learn more about the Azure Marketplace, see Zscaler Private Access Connector (https://azuremarketplace.microsoft.com/en-us/marketplace/apps/zscaler.zscaler-private-access?tab=Overview).

App Connector Placement in Azure

Deploying App Connectors in Azure should occur as close to your applications as possible. As with any cloud service, you do not want to introduce additional latency. Two common deployment models are:

1. Deploying the App Connectors in a VNet with your application instances.
2. Deploying the App Connectors in a gateway VNet that serves other private application VNets.

Deploying App Connectors with Your Application Instances

In this model, you deploy App Connectors alongside your application instances in the same VNet. The App Connectors exist in the same subnet as the applications they service, and all applications and connectors are subject to the same VNet availability.

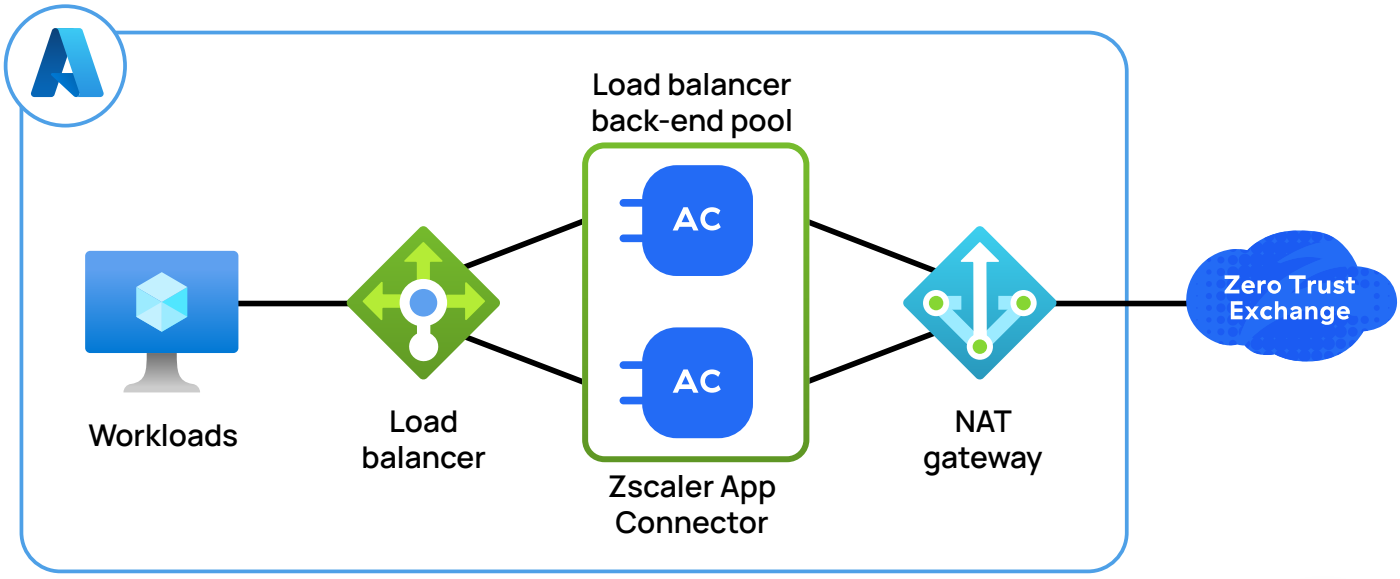


Figure 11. App Connectors deployed with the applications in the same VNet

Gateway VPCs

In this model, you might have multiple VNets running with different types of applications and services across different private VNets. All these VNets share a common gateway VNet that handles their traffic to the internet. For resilience, you still need to deploy connectors across more than one Availability Zone.

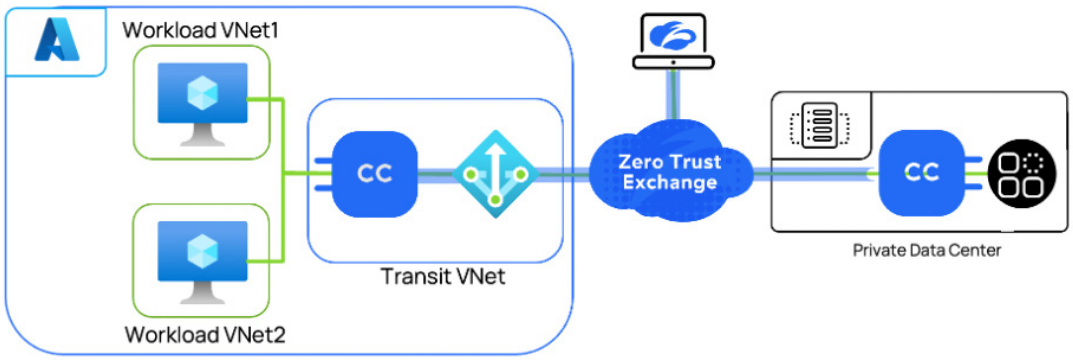


Figure 12. Multiple private VNets using a common gateway

Availability Zone Redundancy

Availability Zones in Azure are Azure data centers in the same geographic region, with discrete power and data connections. There are often multiple Availability Zones per region. The data centers are interconnected with high-speed links as a part of the Azure backbone. This allows you to spread your App Connectors across Availability Zones in the same region, so if one zone should become unavailable, the user session fails across to the App Connectors in another Availability Zone. To view a list of available Azure availability zones, see Azure geographies (https://azure.microsoft.com/en-us/global-infrastructure/geographies/#geographies).

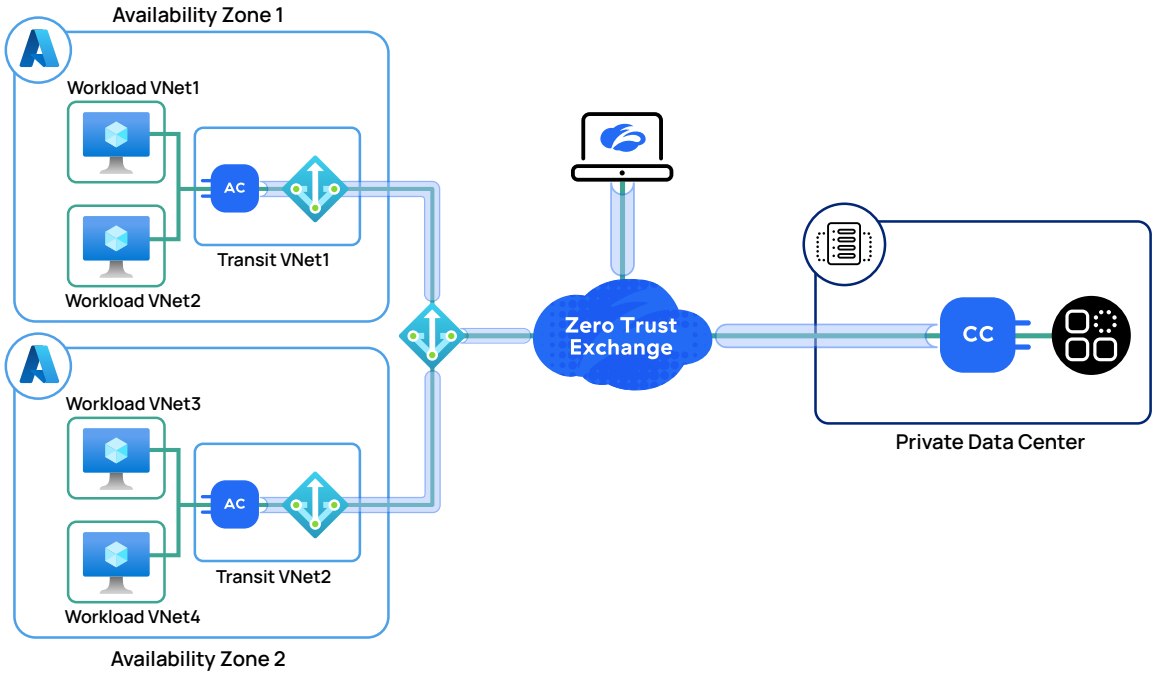


Figure 13. App Connectors in multiple Azure Availability Zones within a region

The Zscaler best practice is to deploy your App Connectors across Availability Zones for increased availability and redundancy. To learn more about regions and availability zones, see Regions and availability zones (https://docs.microsoft.com/en-us/azure/availability-zones/az-overview).

Deployment Recommendations

Zscaler recommends deploying your App Connectors near your applications and internet gateway so that you do not introduce additional latency. Zscaler's best practice is to deploy App Connectors in groups of two or more at every VPC where your internet access is located.

Grouping these App Connectors by region allows for local failover, as well as proper routing by the ZPA system. As an example, if you were in Portland, Oregon, it makes more sense to route you to a US West or US Central data center instead of one in US East.

When considering where your App Connectors need to be placed and grouped, it's ideal to have a map of existing applications. In the development of the policy phase, you'll need to organize your applications and connectors. You'll focus on building logical groups servicing the same applications across your Azure deployment. You might already have some of this information, and you will uncover the rest in the discovery phase.

App Connector Connectivity to the Zscaler Cloud

App Connectors provide access to your users via their connection to the Zscaler cloud. To function properly, connectors need to be able to reach both the Zscaler cloud and your internal applications. You will need to verify the following:

1. App Connectors can reach all Zscaler data centers.
2. App Connector outbound traffic must not be subject to any forms of inline or man-in-the-middle TLS interception or inspection. Zscaler recommends bypassing outbound traffic out of any outbound proxy entirely to facilitate ZPA traffic optimization.
3. App Connectors can communicate bidirectionally with internal applications without ACLs or firewall interference.

To learn more about App Connector connectivity requirements, see **Firewall Requirements and Interoperability Guidelines** (**https://help.zscaler.com/zpa/connector-deployment-prerequisites#SecurityAndFirewallReqs**).

When deploying App Connectors, it is critical to set up the ability to monitor logs. Zscaler and Microsoft recommend using Azure Log Analytics, part of Azure Monitor, to monitor access logs. Log Analytics provides logs from your Security Groups and Access Control Lists (ACLs). Additionally, check any ACLs on the applications themselves for issues. You might need to adjust your ACLs to allow connections coming from the App Connectors.

To learn more about Azure Log Analytics, see **Overview of Log Analytics in Azure Monitor** (**https://docs.microsoft.com/en-us/azure/azure-monitor/logs/log-analytics-overview**).

App Connector Network Connectivity

Azure supports several connectivity models for applications. These range from connecting directly via the internet to having your applications behind one or more virtual network devices.

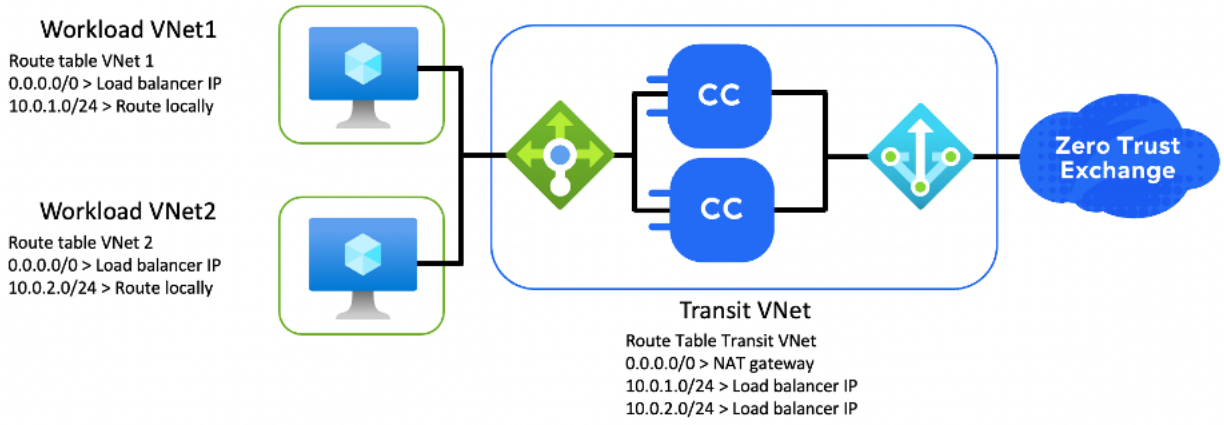


Figure 14. App Connectors behind a NAT gateway in a transit VNet with example IP addressing

Zscaler's recommendation is to keep App Connectors behind a NAT gateway where they can directly reach your private applications. Azure recommends that a NAT gateway be placed behind an internet gateway.

Zscaler's recommendation is to configure 0.0.0.0/0 outbound from App Connectors. However, if you are required to reduce the outbound connectivity of your instances, you'll need to ensure access to all ZPA Service Edge IP addresses. Remember that the Zscaler App Connector only makes outbound connections, so multiple App Connectors behind a NAT Gateway is the correct deployment to avoid any need for inbound connections.

While supported, Zscaler does not recommend placing the App Connector directly on the internet with publicly addressable IP. Like any computing platform, you should take a defense-in-depth strategy to securing App Connectors from direct internet access.

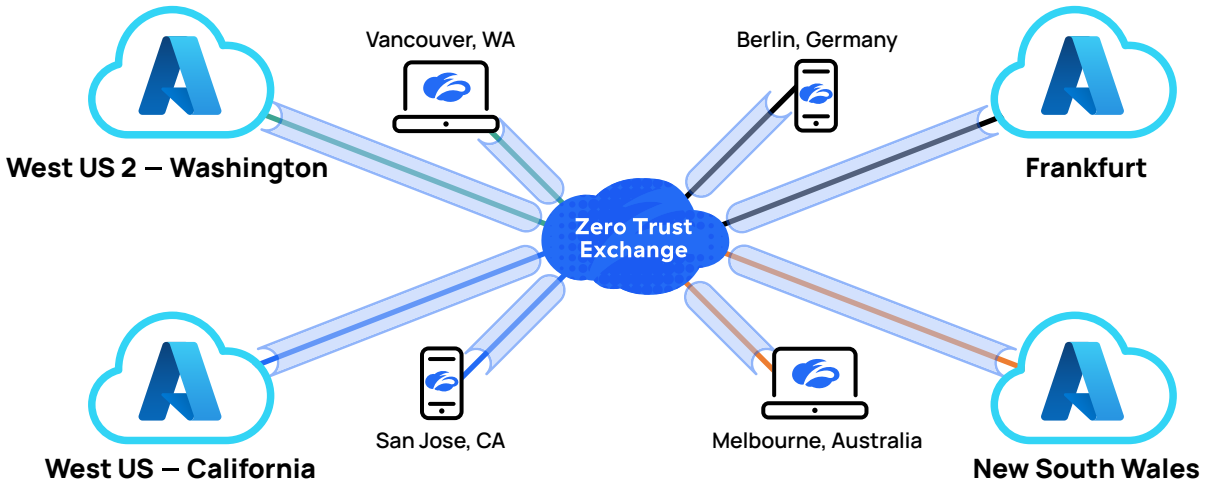


Figure 15. Users connect to their closest data center

It's also important to remember that your users move. They won't always be connected to the same ZPA Service Edge instance. For example, when users travel overseas, they will connect to the nearest ZPA Service Edge. Even when at home, an internet outage by a provider can lead to users connecting to a different Service Edge. To support this movement, all App Connectors must be able to establish a connection to any Zscaler data center where a ZPA Service Edge is deployed.

All IP ranges are published online, and Zscaler provides expanded ranges with 90 days' notice. You can view the ZPA Service Edge IP ranges at Zscaler Private Access Firewall Whitelist (https://config.zscaler.com/private.zscaler.com/zpa).

To learn more about Azure deployment models using a NAT gateway, see Design virtual networks with NAT gateway (https://docs.microsoft.com/en-us/azure/virtual-network/nat-gateway/nat-gateway-resource).

Application Traffic

The App Connector becomes the gateway by which your users can reach your internal applications. The App Connector and the application need to be able to communicate without being blocked by internal firewalls or inspections. You will likely need to adjust firewall and ACL rules to ensure any existing security protections are updated. This can include:

- Any firewalls sitting between your application and the App Connectors
- Any server or application-based access lists
- Networking and routing configurations on the individual servers and/or applications

Server-to-client traffic is not supported. The App Connector only responds to requests from authorized clients.

ZPA carries client-to-server TCP, UDP, and ICMP traffic between users and applications. These connections are always brokered by the ZPA service to establish the connection. This is a model where the client requests and the server responds. However, when a connection is established, bidirectional communication within the session occurs as usual.

App Connector Software Updates

Maintenance of App Connectors is an automated process handled by Zscaler and is configured at the group level. You need to specify a date and time for the group updates.

Updates occur in a four-hour upgrade window and, during that time, a randomly chosen App Connector within an App Connector group performs the update process. When the App Connector is back online, another member of the group is chosen. This process continues until all App Connectors in the group have been successfully updated or the maintenance window has expired.

Zscaler recommends keeping your App Connector groups small enough to remain in one geographic region's maintenance window. App Connectors are required to be deployed in pairs so that the upgrade process does not remove all access to the applications being served.

To learn how to configure App Connector updates, see Scheduling Periodic Software Updates for an App Connector Group (https://help.zscaler.com/zpa/scheduling-periodic-software-updates-connector-group).

App Connector Host OS Security

Zscaler App Connectors run on the Linux operating system (currently CentOS 7.2). The images provided by Zscaler only enable the minimum required listening services. The VMs are updated regularly to reflect the latest Zscaler product updates.

Zscaler handles the maintenance of the App Connector software in maintenance windows that you configure. It is your organization's responsibility to maintain the host OS where the App Connector software resides. See the following section on maintaining the host OS software.

Host OS Software Updates

The host OS software needs to be maintained as a part of your standard operations and maintenance. There are two primary methods for updating the host OS:

1. Temporarily disabling the App Connector and performing a host upgrade using the YUM package manager.
2. Destroying and recreating the App Connector with a newer image.

It's a Zscaler best practice to update the connectors in place as it results in the least overhead. This maintenance should be performed at the same interval you would use for your other Linux OS hosts. The procedure for upgrading the host OS can be found at Upgrading the App Connector Host OS (https://help.zscaler.com/zpa/upgrading-app-connector-host-os).

There might be cases where the host is not upgrading properly, or you are concerned that the host has been compromised in some way. In these cases, Zscaler recommends building a new host and disabling the current host. You can find the procedure for replacing an existing App Connector at Replace a App Connector Using an Existing App Connector Provisioning Key (https://help.zscaler.com/zpa/managing-deployed-connectors#RefreshConnectorExistingKey).

Designing for Redundancy

Setting up redundancy with App Connectors is a straightforward planning exercise. Redundancy is configured at the VPC level and extends across VPCs as you add more App Connector groups with access to the same applications. Zscaler recommends spreading your connectors and applications across Availability Zones in Azure to prevent localized outages.

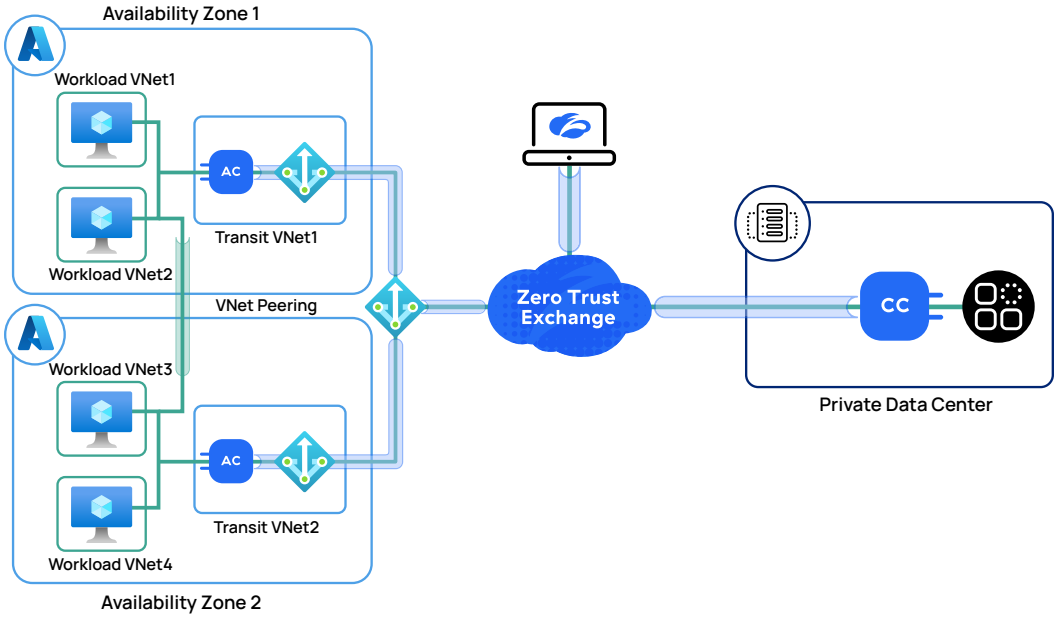


Figure 16. App Connectors in multiple Azure Availability Zones within a region

App Connector Redundancy

Zscaler licenses App Connectors as pairs. You must always deploy App Connectors in groups of two or more for high availability. You might also need to deploy additional instances for capacity reasons.

Single App Connectors are a deployment risk. Any service interruption takes the application offline if no other App Connectors can reach it. This is also true when you perform upgrades.

Virtual Network (VNet) Redundancy

If your application exists in more than one VNet, the App Connector groups for each of the VNets should be associated with the application (via Application Segment and Server Group mapping; see **Application Segments** later in this guide). This allows the ZPA service to connect the user through the nearest data center. If a data center goes offline, connections automatically fail over to the next closest VNet hosting the application.

If you have an application available in several different regions, Zscaler recommends creating a connector group for each region. Deploy your App Connectors into each region, placing connectors from each region in their own connector groups. From there, you associate all App Connector groups with your application group. For more on planning App Connector groups, see **App Connectors** earlier in this guide.

Do not create a single connector group and place App Connectors from that group in multiple Azure regions. If you do so, you will not have deterministic connectivity or appropriate failover.

Azure backend routing and VNet redundancy are beyond the scope of this guide. To learn more, see **About zone-redundant virtual network gateways in Azure Availability Zones** (**https://docs.microsoft.com/en-us/azure/vpn-gateway/about-zone-redundant-vnet-gateways**).

Deploying App Connector via Scripts

App Connector can be deployed directly from the Azure Marketplace, or through ARM Template scripts or Terraform scripts. While supported, deploying App Connector manually is not best practice outside of lab environments. Zscaler recommends using scripting to provide consistent deployments across your organization. ARM Template scripts in Azure are more "native" to their respective platforms, however the preferred method for deploying the appliances is via Terraform.

Deploying App Connector via Terraform

Zscaler Terraform scripts provide complete end-to-end automation to not only instantiate the App Connector appliances, but all the secondary and tertiary components as well (in a best practice configuration). Terraform scripts can be downloaded from the App Connector portal in two versions:

- Starter Deployment Template – Instantiate a Resource Group containing App Connector appliance, VNet, Route Tables, Subnet, NAT Gateway, and Network Security Groups for use cases where only ZIA is required. In addition, Terraform also creates a VM instance for use as a Management/Bastion host in the VNet that App Connector is deployed in. This host is not a requirement long term, but is recommended for easier troubleshooting and testing.
- Starter Deployment Template with Load Balancer – Instantiate App Connector in high-availability mode using Azure Load Balancer, along with required cloud constructs. In addition, Terraform also creates a VM instance for use as a Management/Bastion host in the VNet that App Connector is deployed in. This host is not a requirement long term, but is recommended for easier troubleshooting and testing. To learn more, see **Azure Load Balancer** (**https://azure.microsoft.com/en-us/services/load-balancer/**).

It is important to note that Terraform does not modify brownfield deployments. When executing Terraform scripts, new VNets, Route Tables, Subnets, and VM instances are spawned to support the current workflow. It is the customer's responsibility to integrate the new deployment into their existing environment. This can mean that the new App Connector VNet is peered with existing VNets, or that new workloads are installed within the App Connector VNet. Bear this in mind when considering whether Terraform is the correct option to use when integrating with a brownfield environment.

For detailed deployment instructions and to find the templates listed above, see **About Cloud Automation Scripts** (**https://help.zscaler.com/cloud-connector/about-cloud-automation-scripts**).

Deploying App Connector via ARM Template

For customers seeking a more native automation option for deploying App Connector, Zscaler offers ARM Templates. Though ARM Templates can be used in greenfield situations, their value shines when a customer is seeking brownfield integration, since many of the prerequisites are already satisfied if a customer has an existing Azure buildout.

It should be noted that although ARM Templates scripts work well with brownfield deployments, it is still your responsibility to integrate them into the environment.

For detailed deployment instructions and to find the templates listed above, see **About Cloud Automation Scripts** (**https://help.zscaler.com/cloud-connector/about-cloud-automation-scripts**).

Leveraging Azure Autoscale for Redundancy

The Azure service uses the concept of Virtual Machine Scale Sets, which allows you to set a minimum, desired, and maximum number of instances of your application. These groups allow you to configure the number of instances you need for your average connection load, automatically bursting when needed. This ability allows you to purchase long-term reserved instances for your baseline capacity, lowering costs. You can also be ready to expand automatically should something unexpected occur.

Virtual Machine Scale Sets are optional, and not required for a ZPA deployment.

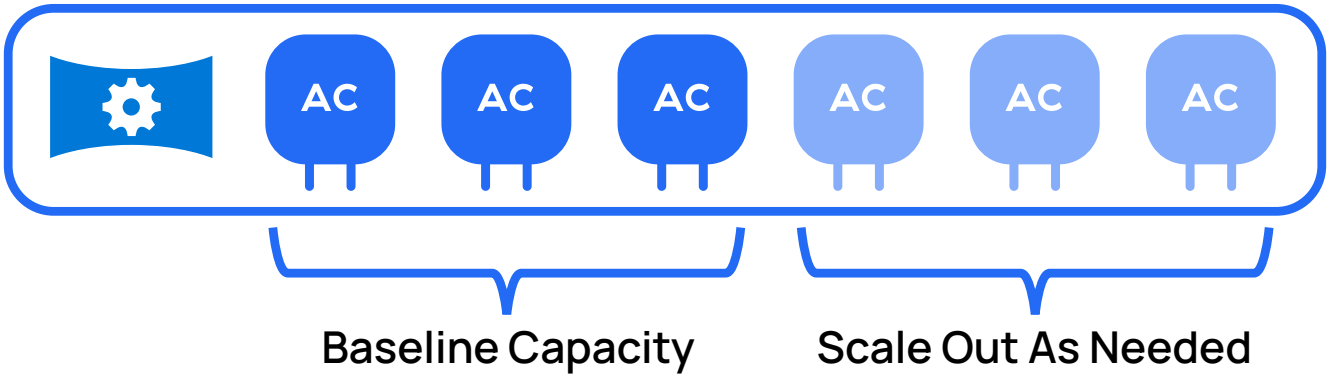


Figure 17. Scale out App Connector instances to meet surprise spikes in demand

When planning your Virtual Machine Scale Sets, you should keep in mind that autoscale is not instantaneous. Triggering a new App Connector instance to come online still requires that instance to spin up, connect to the Zscaler cloud, perform any updates, and finally start serving clients. As such, Zscaler does not recommend Azure Autoscale as a cost-savings method.

Instead, you should think about Azure Autoscale as a form of standby capacity and redundancy. For example, if you know your users working from your headquarters always hit the same App Connector instances, you can scale those to meet your average daily load. In addition, you can build an Azure Autoscale to account for any large temporary influxes of users, such as company meetings or customer events you host.

Another reason for building Azure Autoscale for redundancy is to handle outages. Should access to a set of App Connectors become unavailable due to an outage, users will fail over to the next nearest data center. If those App Connectors become saturated, your ability to launch a new instance to scale up automatically can help mitigate load issues.

When you no longer need the extra instances, the Azure Autoscale can terminate the instance automatically. However, this can cause a temporary degradation for users who are connected to that instance, as it shuts down without waiting for users to necessarily migrate to other instances. Instead, Zscaler recommends setting up a termination policy within the Azure Autoscale to prevent the new instance from automatically turning itself down.

Azure Autoscale supports notifications via email and webhooks, allowing you to configure alerts when your load has decreased on the new instance. You can then go through the process of manually turning down the connector instance that was brought up with Autoscale and gracefully move users to the remaining App Connectors on reserved instances.

By leveraging launch templates, you can allow Azure to add capacity to your deployment automatically. When setting your values, Zscaler recommends setting your App Connector group to a minimum and desired value of at least two instances for redundancy purposes.

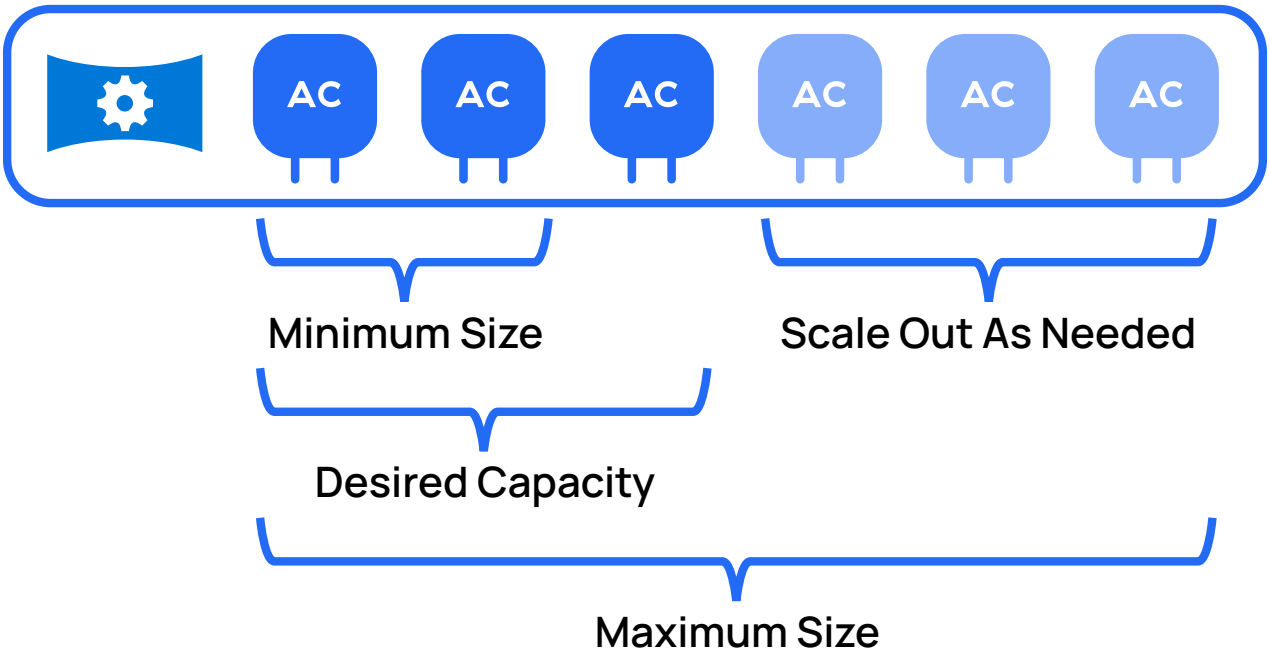


Figure 18. Azure Autoscale of App Connectors

You should set the minimum scale size to match your expected daily load. These would be your minimum instances. You should set the maximum large enough to account for any expected burst load.

Be sure to configure your App Connector provisioning key to allow for the same level of maximum instances that are in the Azure Autoscale. If the provisioning key maximum is lower than the Azure Autoscale maximum, new App Connectors will fail to launch successfully.

As mentioned previously, you can also leverage Autoscale groups as a method to simplify App Connector updates. By destroying the instances, the Autoscale group will rebuild them from the latest Zscaler image. This allows you to bypass host OS patching.

To learn more about Azure Autoscale, see Overview of Azure Autoscale in Microsoft Azure (https://docs.microsoft.com/en-us/azure/azure-monitor/autoscale/autoscale-overview).

To learn more about Azure Autoscale notifications, see Use Autoscale actions to send email and webhook alert notifications in Azure Monitor (https://docs.microsoft.com/en-us/azure/azure-monitor/autoscale/autoscale-webhook-email).

Creating and Evolving Access Policy

The ZPA application access policy is a flexible set of profiles that combine to allow you to define applications and services available to your users. These applications can be open to all users, or limited sets of users, based on authentication criteria and other context requirements. The system allows you to define multiple access policies, and users can match different policies depending on your business requirements.

Integrating ZPA and Your IdP

ZPA, built on a zero trust architecture, is based on the principle of least privileged access, which involves proving identity and meeting required context to access resources. To build the identity assertion, Zscaler leverages your existing IdP. Authentication and context are communicated via Security Assertion Markup Language (SAML) and used by ZPA to authorize users of the system.

When a user requests an application, the SAML assertion is parsed. Based on the policy you've defined, the user has an access policy applied to their traffic. The access policy defines a user's access to applications.

By leveraging your existing IdP, you don't need to maintain user accounts in multiple places; users are authorized against your existing system. By using System for Cross-Domain Identity Management (SCIM), the system can also modify or remove a user's authorization automatically.

One of the other benefits of SAML is that it can return any number of attributes in an assertion. Some examples that your organization might already use include:

- Directory attributes, such as a group or department
- Device attributes, such as whether it's a managed or unmanaged asset
- Authentication attributes, such as certificate-based authentication vs. username and password

Based on the factors you include, the user's current posture, organization role, and the type of traffic are matched to policies. Check with your identity and access management team to understand which attributes have been populated by your organization's IdP. To learn about importing your SAML attributes for use by ZPA, see **About SAML Attributes** (**https://help.zscaler.com/zpa/about-saml-attributes**).

Learn more **About Identity Providers** (**https://help.zscaler.com/zia/about-identity-providers**).

Learn more **About SAML** (**https://help.zscaler.com/zia/about-saml**).

Learn more **About SCIM** (**https://help.zscaler.com/zia/about-scim**).

IdP Configuration Help

Setting up your IdP to work with Zscaler is a critical first step in configuring ZPA. You need to configure both Zscaler and your IdP to work together as follows:

- On the Zscaler side, configure your IdP. To learn more, see **About IdP Configuration** (**https://help.zscaler.com/zpa/about-idp-configuration**).
- On your IdP side, configure Zscaler as the service provider (SP) or Relying Party Trust on Windows Server.

Legacy Active Directory Servers

Like most applications, Active Directory has evolved over the years. There might be some legacy domain access for applications and tools running that require an older authentication method. You might also find that there are more subdomains still in existence than you had originally understood. It's always best to check with your identity and access management team to find out exactly what's still running and what apps and tools might still be in use.

The need for legacy domain access can affect Zscaler's wildcard access policy. If you find out that more domains are still in use through the system, you'll need to modify your auto-discover process to take those domains into account.

Access Policy Decision Criteria

An access policy is where you define user-based access control. It is a combination of defined users and application segments or application groups that the users can access. This is the policy where you ultimately make decisions about how your users access the system.

When a user successfully authenticates, the returned attributes from your IdP are used to make a policy selection for the user. That policy determines which applications the user can find and access, and can differ depending on how, when, or where the user has requested access. For each policy, the user is allowed or denied access to resources.

As mentioned previously, the user's policy can change depending on several factors, such as device, time of day, and location. This is different from role-based access control, as users are not granted a static role. They are instead given a set of policies that are appropriate for their current authentication state.

In addition to access policies, timeout policies can be used to control how frequently a user must reauthenticate to access applications. As with access policies, timeout policies can be configured based on combinations of multiple context criteria.

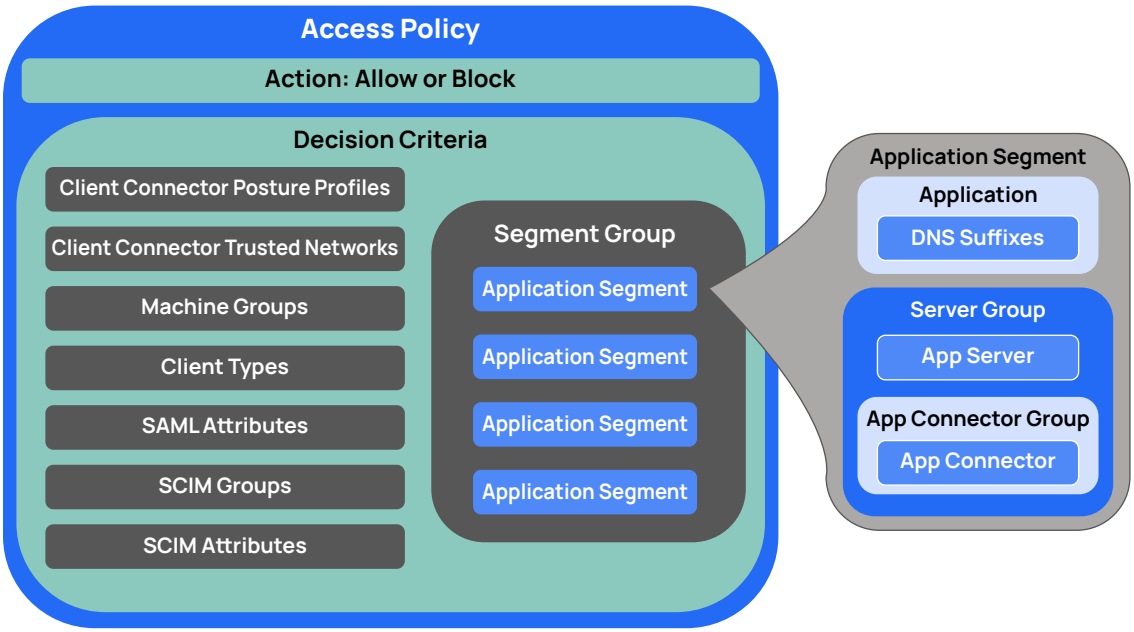


Figure 19. Access policy decision criteria

The following lists include the profiles in the system and how they are used. We cover each in more detail in the following sections.

User and Device-Based Criteria

- Application segments and/or segment groups
  - Application segments – A grouping of defined applications based upon access type or user privileges.
  - Segment groups – The group of configured application segments.
- Client Connector posture profiles – The set of criteria that a user's device must meet to access applications with ZPA, observed by Zscaler Client Connector.
- Client Connector trusted networks – Known networks, defined by the administrator, that Zscaler Client Connector detects that the endpoint is connected to.
- Client types – End-user traffic originates from Zscaler Client Connector, machine tunnel, or web browser. Other traffic might originate from Zscaler App Connector or ZIA Service Edge.
- Machine groups – The group of configured machines (used for pre-Windows login access by the user's endpoint).
- SAML attributes – User attributes obtained via the SAML assertion from an IdP.
- SCIM attributes – User attributes learned from an IdP.
- SCIM groups – SCIM groups learned via SCIM from an IdP.

Applications

- Applications – An application is defined as a fully qualified domain name (FQDN), IP address, wildcard subdomain, or IP subnets accessed via a standard set of ports.
- Application segment – A grouping of defined applications based upon access type or user privileges on a shared set of access ports. Note that an application can only be a part of one application segment, so plan your application access carefully.
- Application segment group – Application segment groups allow you to combine different application segments into a single group and then apply policy to the group.
- Application server – Either a manually defined (not recommended) or dynamically discovered application server that hosts an application.
- Server group – A group of application servers/instances that serve a given application. The server group maps the application segment to the App Connector group(s) with access to the Virtual Network (VNet) where the servers in the server group are located.

App Connectors

- Provisioning keys – Used to authorize an App Connector to connect to your tenant and assign it to an App Connector group. When you bring up a new App Connector, the provisioning key is added to the configuration.
- App Connector(s) – Virtual machine (VM) instances that provide access to applications in combination with the ZPA Service Edge. Zscaler provides App Connector appliances in the Azure Marketplace.
- App Connector group – A logical group of deployed App Connectors, with each of the App Connectors existing in a specific group. This group is also tied to a provisioning key and a server group. App Connectors in a group must have access to the same applications and should reside in the same VNet.

Policy Definition Framework

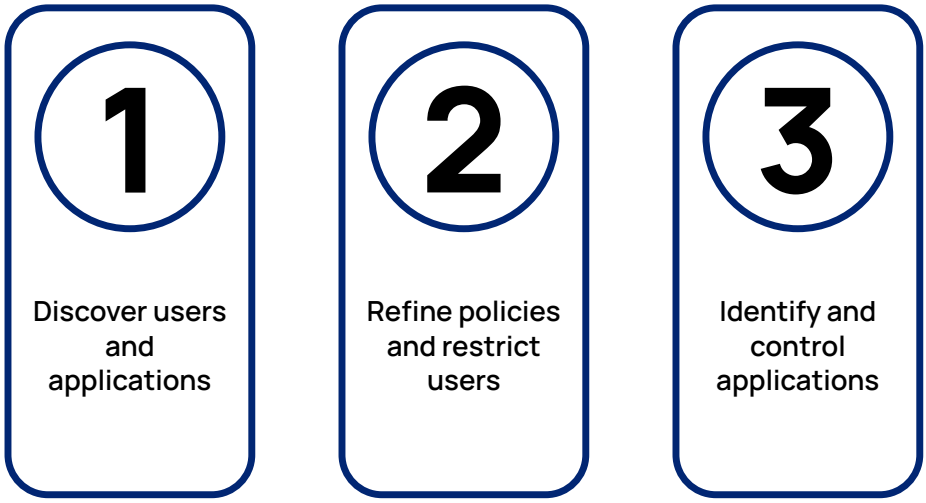


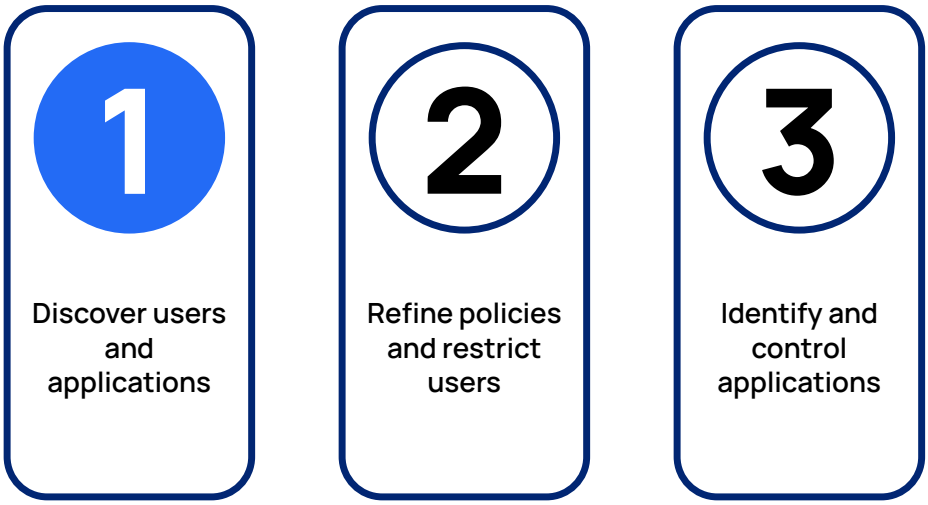
Figure 20. Policy definition framework

When defining policy, you should take a phased approach from discovery to control.

In Phase 1, start by bringing over users in groups and watch what applications are accessed and by whom. In Phase 2, start to restrict your most critical applications while continuing to bring users on board. In Phase 3, you can wrap up your policy and make additional refinements.

Zscaler recommends this approach instead of simply trying to re-create an existing access policy. By taking a fresh look at how users interact with applications, you can make better policy choices and stop relying on legacy decisions.

Phase 1 – User and Application Discovery



Building a policy is a process that often starts with discovery. To provide appropriate controls, you need to understand what applications are in use and by which users. While you likely know the major applications, you might not know exactly who should have access to them. There will also be a great number of applications that exist for a particular group or function that might not be clearly understood.

While it's important to understand your current remote access system and policy, we don't recommend relying on it to provide a current state of your network. Networks and applications grow and evolve over time, people change roles, and companies merge and divest units. Often, legacy rulesets exist past their useful time.

Instead, Zscaler recommends taking a discovery approach to your policy design, informed by workflows that exist today between your users and applications. To learn those workflows requires direct observation, and that's where you should start with your ZPA policy.

Before you begin, note that the policy you build in this step allows all users to see all of the applications where applied. This is the goal of discovery: to gather information to identify applications and build policy for them. The Zscaler best practice is to use a controlled rollout of this policy to select groups of users. This gives your team the time to identify and classify apps and users in batches. As more groups are added, you only need to classify new applications.

First Rule Match

Superficially, ZPA access policy rules are similar to a stateful firewall rule set in that they control access between a source and destination and are applied top-down, first-match. Unlike a firewall policy, a more specific domain takes precedence for matching over a wildcard domain (and a specific IP takes precedence over an IP subnet), so a policy applied to a more specific target is matched rather than a policy for a less specific target, even if the policy for the less specific target is placed higher on the list.

A traditional firewall rule is selected exclusively on the first match as you fall through the policy list. The list is made up of the standard 5-tuple being: source IP address, source port, destination IP address, destination port, and the transport protocol. But as we've seen, ZPA provides another option in wildcard matches.

Wildcard matches are treated as less specific than an FQDN. If an FQDN matches exactly, that is considered more specific than the wildcard, and that FQDN will no longer match the wildcard app segment or any access policies that refer to it. Zscaler recommends following the standard convention of putting more specific policies first, as that is easier for your administrators to understand.

For example, we'll look at rules configured and ordered as follows:

Dest. Application	Criteria	Action
*.safemarch.com	TCP: 22	Deny
dev.safemarch.com	TCP: 22	Allow

Let's assume this policy set is applied to a developer who should have SSH access to the dev environment. If the developer launches an SSH session to *finance.safemarch.com*, the connection will be denied, as it matches the first rule. In this case, there is no specific rule allowing or denying *finance.safemarch.com*, so the first rule matches as it's the most specific rule.

If the developer then launches an SSH session to *dev.safemarch.com*, it will be allowed, even though it comes later in the list. In this case, the request matches the second rule because the FQDN is more specific than the generic wildcard, so that rule takes precedence.

Wildcard matches are considered less specific than fully specified hostnames or FQDNs, even if the wildcard comes first in the policy rule set. The Zscaler best practice is to keep your wildcard policies at the bottom of your rule set to help future admins understand the policy flow without having to view the documentation. Without this practice, you can cause future admins frustration when troubleshooting access policy issues, particularly if they have experience with traditional firewall appliances.

ZPA Boolean Operands

ZPA gets its policy flexibility and power from the ability to specify multiple attributes and use Boolean operands to match multiple criteria:

- AND – Must match all conditions to be true.
- OR – Can match either condition and be true.

These operands are mutually exclusive. If you need to have two sets of AND criteria, you would have to build two rules. For example, if you wanted your deployment specialists to be able to SSH into your dev environment, but only while at the corporate office in either San Jose or Chandigarh, you would build two rules.

Dest. Application	Criteria	Action
dev.safemarch.com	IdP Role: DevOps - AND - Location: SJC	Allow
dev.safemarch.com	IdP Role: DevOps - AND - Location: IXC	Allow
dev.safemarch.com	IdP Role: DevOps	Deny

ZPA does **not** currently support nesting rule sets, such as:

IdP Role: DevOps - AND – (Location: SJC - OR - Location: IXC)

For each context criteria, your policy must use only one of the two operators. They will be linked all the way through the policy.

Building a Policy for Application Discovery

An initial wildcard policy does not apply enforcement. Instead, it provides you with the necessary pieces of information for you to begin to form policy:

- What services are on your network, and does this information agree with your understanding of the environment?
- What users are leveraging the services, and are those the appropriate users or groups?
- What applications exist that you didn't know existed, such as a process that's effectively been forgotten about or an instance of shadow IT?
- What applications are not reachable by all of your App Connectors? This might include applications deployed in only a limited number of data centers or an existing physical data center.

To gather this data, you need to allow what is already happening to continue—at least initially. Your app segment for application discovery will simply be a wildcard to match everything in an application domain or subdomain, as expressed below:

*.[yourdomain].[tld]

TCP Ports 1-52 & 54-65535

UDP Ports 1-52 & 54-65535

Following the creation of your wildcard, create an access policy allowing your group of pilot users to access that wildcard app segment. Since everything is allowed, nothing should break. This is a critical step in gaining user trust for the new solution.

For your applications, it is critical that you avoid UDP 53. ZPA has special handling for DNS requests, so we cannot have applications also advertising DNS records through the service.

As users access applications, you'll see their activity on the ZPA user and application dashboards. You'll gain a solid understanding of applications and services running, how they are being accessed, and by whom. With the application-user interaction map, you can define more granular policies in the next phase.

To learn more about specific dashboards and functions, see About the Applications Dashboard (https://help.zscaler.com/zpa/about-applications-dashboard).

Domain Suffixes and Search Domains

Most modern apps have evolved to use FQDNs given the nature of virtual machines, containers, and the cloud. However, there are numerous legacy apps that rely on short names and the suffixes provided by the DHCP server. You need to understand these applications to provide consistent policy.

In your discovery planning, it's useful to review which DNS suffixes and search domains are in use across your organization. Looking at configurations for DHCP servers can help you understand the complete picture of search domains in your organization.

You need to replicate these domain suffixes by defining search domains in ZPA to go along with the wildcard discovery for those applications. With your list ready, you can configure DNS search domains and ensure that Zscaler Client Connector checks your internal server first. To do this, define your DNS search domains by specifying your domain and TLD; no leading * wildcard is required.

[yourdomain].[tld]

It is a best practice to enable "Domain Validation in Zscaler App" in the config. This feature enables Zscaler Client Connector to check your DNS resolution internally before testing against external servers. This is especially critical for organizations with so-called "split-brain" DNS, a DNS deployment style that has both internal and external DNS servers with different records in the same namespace.

Bypass Domains Advertised Both Externally and Internally

When you have specific applications that are advertised both by your internal and external DNS, the Zscaler best practice is to leverage your external DNS. To do so, configure a domain bypass app segment that contains all the externally advertised FQDNs and set the Bypass option in the app segment to Always. This ensures that your users leverage your external DNS for resolution of publicly available services.

In this case, you might advertise the *safemarch.com* domain both internally and externally. You would want to have your policy allow traffic to internal apps in **.safemarch.com*, but you would put in a bypass for your marketing site *www.safemarch.com*, since that is publicly resolvable and should just go through your standard internet access.

The fastest way to get started is to simply see what DNS records are configured for your organization on the public side. You should also consider external applications that resolve internally, such as Skype for Business. You can then create an application segment that holds only those applications.

Handling IP-Only Hosts

Ideally, users would not access applications by IP address, which is far less flexible than a hostname or FQDN. However, some of your applications that were designed for legacy data center architectures might return IP addresses instead of FQDNs, or sometimes both.

The short-term fix is to add the IP addresses of the applications to your access policy. In the long term, you'll want to modernize the application to use an FQDN to simplify your policy and long-term maintenance. When adding the IP addresses of the application, be sure to add its FQDN as well.

Zscaler does not recommend that you configure entire subnet ranges in your policy. Instead, the best practice is to only allow the specific IP addresses for the applications until they can be modified to leverage FQDN. In the limited cases where it might be necessary to define an IP subnet for discovery purposes, we recommend transitioning to individual IP addresses as soon as possible. We also strongly recommend that you configure both the IP address and FQDN of applications that currently return IP addresses to ease your transition.

Handling Applications with Limited Data Center Deployment

It's not uncommon for organizations to have a hybrid deployment, with some applications in the cloud and other legacy applications in the corporate data center. This could be because the legacy application simply hasn't been moved yet, or because it won't be moved due to policy, license, or regulation.

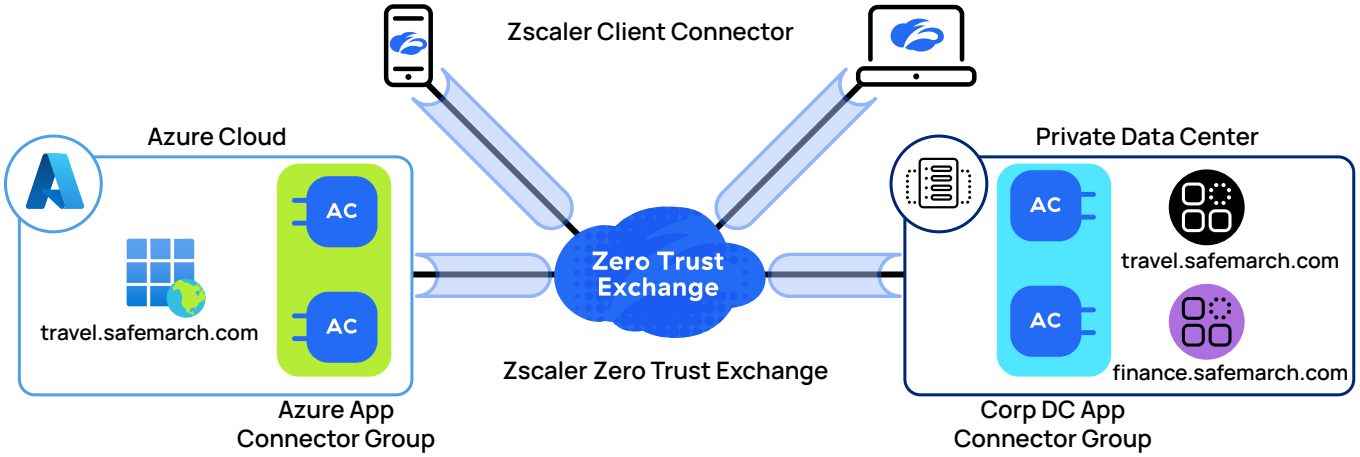


Figure 21. Different App Connector groups have access to different applications

App Connectors for such applications need to be in a separate App Connector group associated with that data center. The reason is that the auto-discover process is built on all connectors having access to all the same applications. When an App Connector serves only a limited number of applications, or applications are only in specific places, those App Connectors need their own group.

Understanding and Resolving Errors on the Dashboard

It is not uncommon for errors to appear on your ZPA dashboard as a routine part of network operations. Remember that ZPA sees every connection coming in from the user, and, due to the nature of networking, not every one of those connections will be successful. Internet applications generally have to negotiate protocols, and that process can fail. Other times, for example, you might be sent a link you don't have permission to access. All of these completely normal operations will generate errors.

What you're seeing is how the applications always worked, but now you have greater visibility into these workings. While it is unlikely that you can ever eliminate 100 percent of errors, that's OK because not all errors are benign.

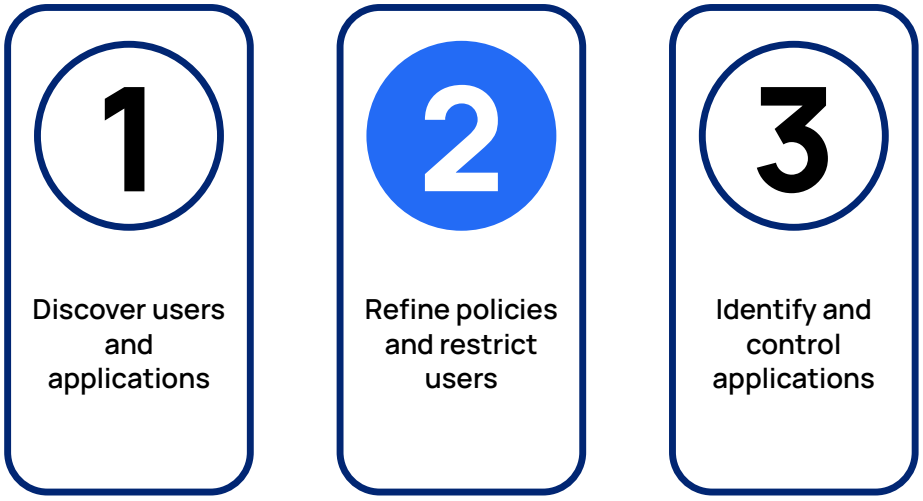
Some errors point to real-world problems and need to be resolved. If one of your groups of users can access an application and another can't, you can start to investigate why. Is the access policy defined correctly? Is there an ACL in the way of one group's App Connector? Is there a network misconfiguration? You might also need to remove that application from that connector. Some applications might need their own app segment with a modified application list.

You can drill into any error and get more details about the failure, who tried to access the application, and what functionality it provides. At this stage, you can decide if this is a real problem that needs to be corrected via a policy or configuration change, or if this is simply something that is accepted as part of how the system works, based on your policy and organizational objectives.

To learn more about user, application, and health dashboards, see Dashboards & Diagnostics (https://help.zscaler.com/zpa/dashboard-diagnostics).

To learn more about ZPA live logs, see About Live Logs (https://help.zscaler.com/zpa/about-live-logs).

Phase 2 – Refine Policies and Restrict Users



Now that you have gathered observational data from our discovery policy, you can compare this to your company security policy for groups and services. Based on the outcomes, you can adjust your configuration for certain applications.

Approaches to Policy Development

As these policies evolve, the wildcard policy will either be modified or removed. Modifications might include factors such as posture checks, locations, etc. Because you've allowed users to access any destination, you'll want to control the source.

There are two primary ways to approach your policy:

- Determine which user groups need access to which applications.
- Determine which applications should allow access from which user groups.

In most organizations, a combination of the two approaches is used.

Determine Policy by User Grouping

In this approach, you start with people: who needs access to which resources? This works well when you are bringing on a set of users, such as through M&A activity in your organization, or through onboarding when a third-party contractor is hired to handle specific tasks.

In both of these cases, you have a very clear idea of the user groups. These communities are specific and well understood in terms of their access. You can allow discovery of apps in some cases, such as M&A. For your contractors, you can allow only their specific application and deny everything else.

For example, let's say you have an HVAC contractor that needs access to the HVAC app at a facility to monitor and manage the air-handling systems. However, the contractor does not need access to any other resource at the site, so the policy should be:

Dest Application	Criteria	Action
hvac.safemarch.com	IdP Role: ext-hvac-tech	Allow
*.safemarch.com	IdP Role: ext-hvac-tech	Deny

This policy allows the contractor to access the necessary resource and nothing more.

Determine Policy by Application Access Needs

This decision takes the opposite approach: a particular resource will be accessed by whom? This is often the approach you take with more sensitive applications. You want to protect those applications and ensure that only privileged users have access.

As you refine your policies, you'll become more granular about specific applications. For instance, you might have a web development server that is in Azure and you write a policy around who can access that application, such as:

Dest. Application	Criteria	Action
dev.safemarch.com	IdP Role: dev	Allow
dev.safemarch.com	IdP Role: *	Deny
*.safemarch.com	IdP Role: *	Allow

In the first rule, you want to ensure that your development team can reach *dev.safemarch.com* to test code. In the second rule, you want to block everyone else from *dev.safemarch.com*, as they aren't in your development team.

Leveraging Policy Criteria

The following image shows the interrelation of various components of policy.

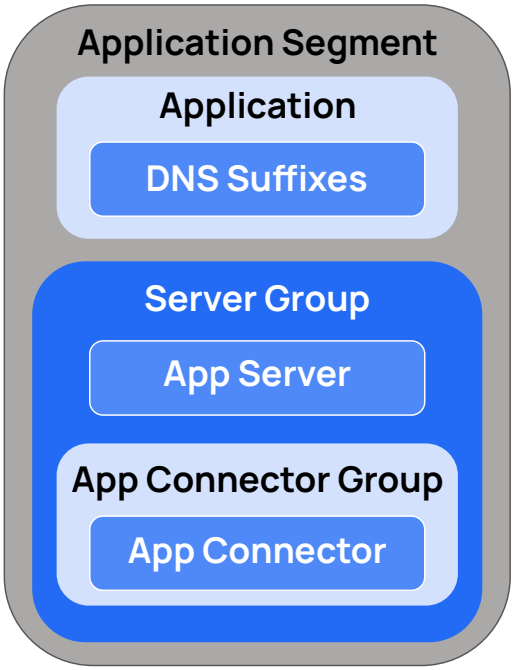


Figure 22. Application segment components

The primary areas of the config you'll use to refine are your policy are:

- Application segments
- Application groups
- DNS suffixes
- App Connector groups
- Server groups
- Application access

Application Segments

Modifying application segments to better reflect the applications they are serving makes your policy creation that much easier to understand. You'll also be able to establish more fine-grained controls over the applications.

If you are using a wildcard application for discovery, be aware that when you define an FQDN, requests for that application will no longer match the wildcard app segment. For example, if you have an app segment containing a wildcard on ports 80/443 for end-user access and port 22 for admin access, as well as an access policy allowing all users to access that app segment, then any user can make both web and SSH connections to the applications in that wildcard domain.

If you later define an app segment containing an FQDN in that domain, mapped to port 22, and create a new access policy to restrict admin access only to admin users, you prevent end users from accessing the web server on ports 80/443. Web requests will not match the new app segment/access policy (it's only for SSH) and will no longer match the wildcard app segment/access policy (because a more specific FQDN app segment now exists). You also need to define a second app segment containing the same FQDN, mapped to ports 80/443, and a corresponding access policy allowing users access that app segment.

In general, any time you define an app segment containing an FQDN that used to be served by a wildcard app segment, you need to make sure that you understand all services supported on that hostname and create appropriate app segments/access polices for each individual service.

Application Groups

Application groups allow you to collect a set of application segments to be used as a single policy decision point. For instance, if your developers use Git to build applications, you can group those in a Dev Apps application group.

DNS Suffixes

DNS suffix changes usually center around a missing lookup or an unexpected return value from Active Directory or something similar. Be on the lookout for DNS suffixes that did not make it onto your initial list and are causing access via application short names to fail.

App Connector Groups

App Connector groups logically group your App Connectors together. This is typically done by location, where all App Connectors are local to each other and have access to the same applications. There can be other factors involved as well, such as which applications the App Connectors are serving. Grouping by location is essential for traffic routing and to ensure upgrades do not occur in the middle of a workday.

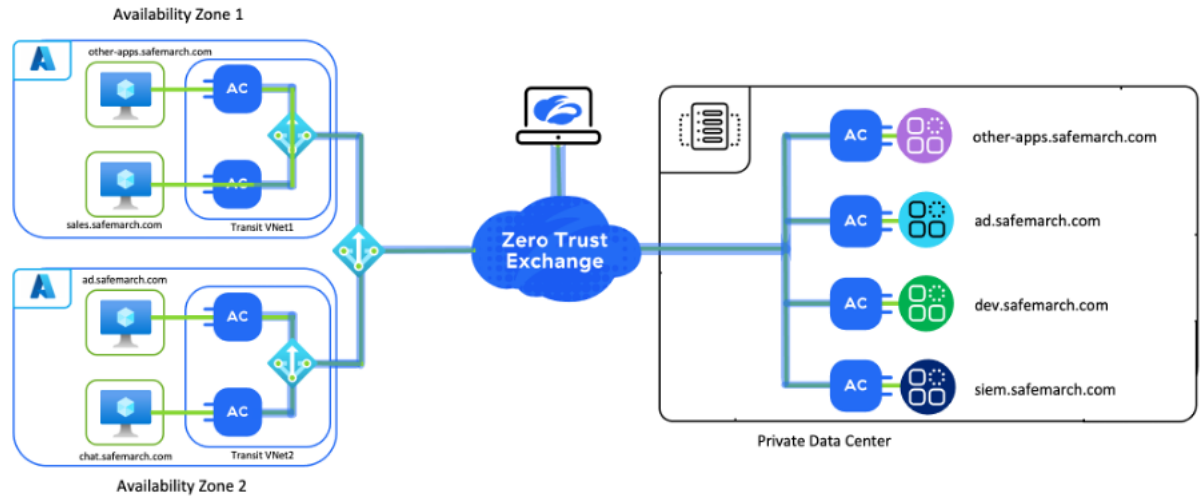


Figure 23. Example network for discussing App Connector groups

The rest of this section covers App Connector groups based on the preceding simplified diagram. For this example, we use a single VNet as a representative example of the rest of the network.

Traffic Routing

When building out your groups, you should seek to contain the groups to a single Azure region or subset of VNets in the same region. Zscaler requires you to input location data for your App Connector groups and, when you do so, the ZPA Service Edge uses that information to steer user traffic to the nearest connector.

When a user attempts to connect to an application, the ZPA service determines the appropriate App Connector group for that connection. If only one App Connector group services an application, it will be the only choice no matter the distance. However, if the ZPA service finds multiple App Connector groups that can access the same application, the group determination will be based on the distance to the user.

When the appropriate group is selected as the nearest App Connector group, a determination is made for the fastest App Connector. Any connector that is within 2 ms of the fastest response is considered equivalent distance, and one of those two connectors is selected based on load.

You should group location-specific applications together in the same App Connector group. Ensure that your users are being connected to the appropriate, geospecific App Connectors. If you place all your App Connectors in a single App Connector group, the performance can suffer, as users might be steered to a more distant App Connector.

Upgrades

Upgrades are scheduled by you for a particular App Connector group. Remember that the Zscaler service picks a random member of an App Connector group to upgrade during the scheduled maintenance window. Because you don't know which member of the App Connector group it will be, it's critical to ensure that all connectors in a group are geographically co-located so that you do not disrupt user work.

Dedicating App Connector Groups to Specific Applications

In most circumstances, a connector handles multiple applications at a location. There are, however, times when you might want to limit App Connector groups to a single application or small group of applications. This situation can occur due to limited deployment of an application, or because the needs of the application itself make a dedicated group advantageous.

Zscaler recommends adding a dedicated App Connector group any time an application begins to starve other applications. This can be bandwidth related, or it could be due to large numbers of ports in use, among other reasons. Creating a dedicated App Connector group with its own App Connectors eliminates resource contention.

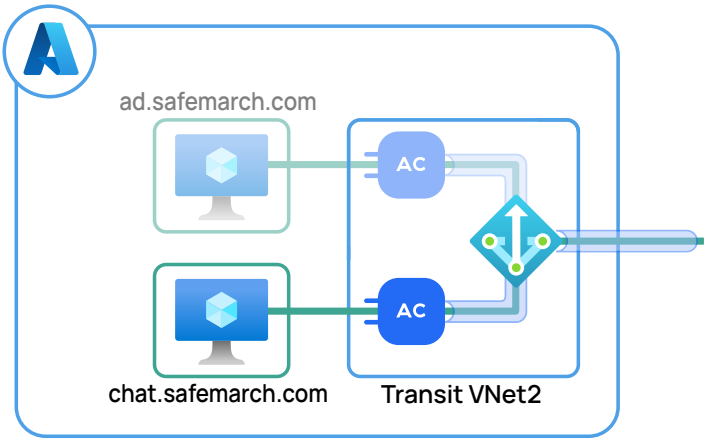


Figure 24. The chat application impacted other applications and was given dedicated resources

In the previous image, the chat application needs a lot of real-time throughput, and it was impacting other applications. The solution was to move it to its own App Connector group with dedicated App Connectors.

App Connector Groups and Active Directory

The design and operation of Active Directory (AD) servers make them especially well suited to having their own App Connector groups. AD servers see a lot of requests as machines discover services and request access. You should ensure that your AD servers are not on the same App Connectors as other applications that might tie up requests for the connectors. Zscaler recommends dedicated App Connector groups for your AD servers.

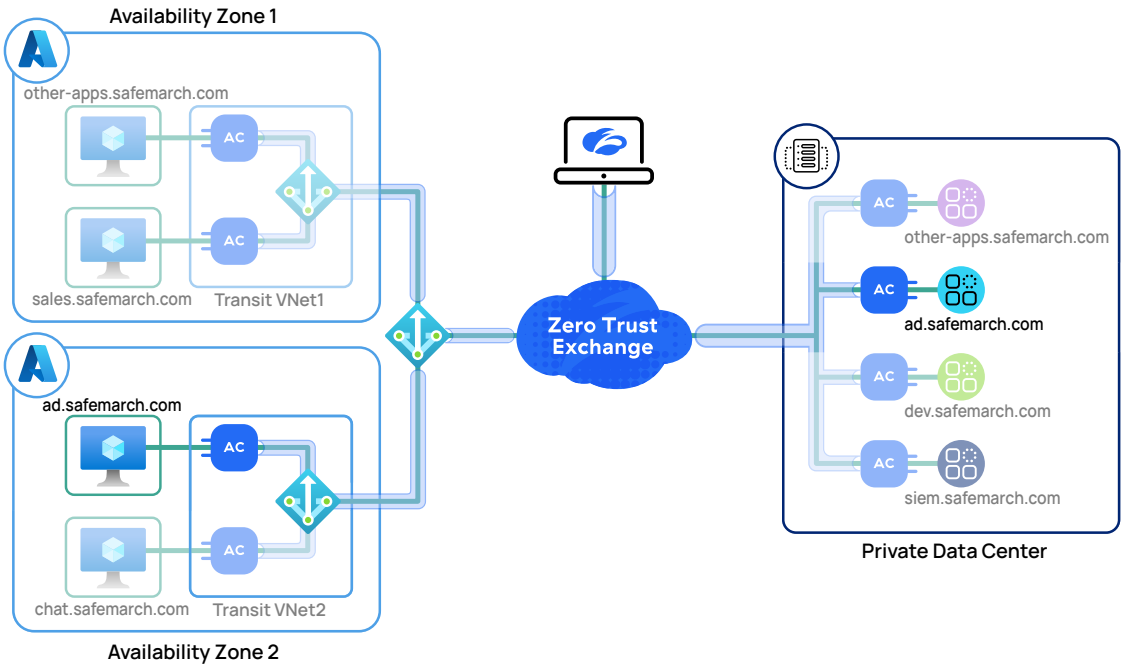


Figure 25. Active Directory with its own App Connector group

The AD service in Azure and at the HQ data center each have a dedicated pair of App Connectors in their own App Connector groups.

AD operates by sending a user request to multiple AD servers at once, and the first response is used. This built-in redundancy means that your AD servers do not need the ZPA service to perform health checks. To learn more about health checks, see Health Reporting.

App Connector Groups and LSS

When using the Zscaler Log Streaming Service (LSS), it's important to place it in its own App Connector group that does not serve user traffic. The LSS is your conduit to move logs from the Zscaler service to your SIEM. Should there be a large spike in user traffic, the log messages might be lost. Zscaler recommends at least two App Connectors in the LSS group, using this group only to service logs streams.

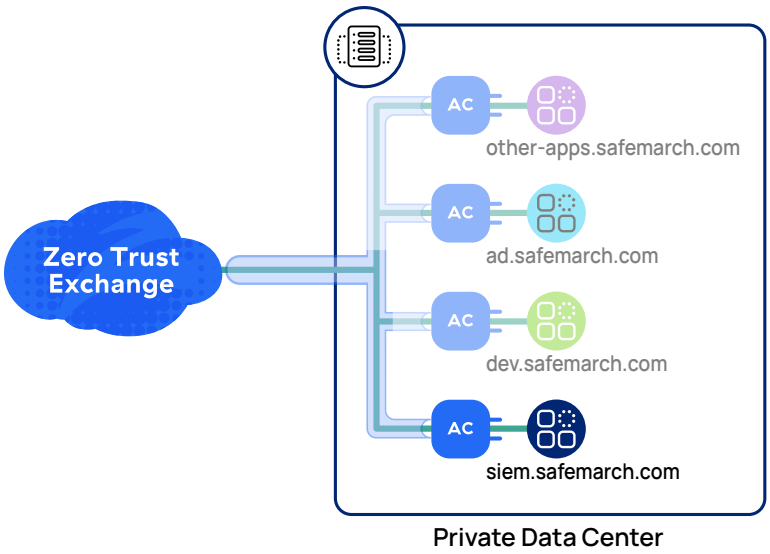


Figure 26. App Connectors for LSS in their own App Connector group

In the previous image, the LSS service is streaming to a pair of dedicated App Connectors. In turn, the logs flow to your SIEM.

Limited Deployment of an Application

For various reasons, such as government or industry regulations, some of your applications might not exist in all locations. When you have an application with a limited deployment, Zscaler recommends placing the App Connectors serving that application in their own connector group.

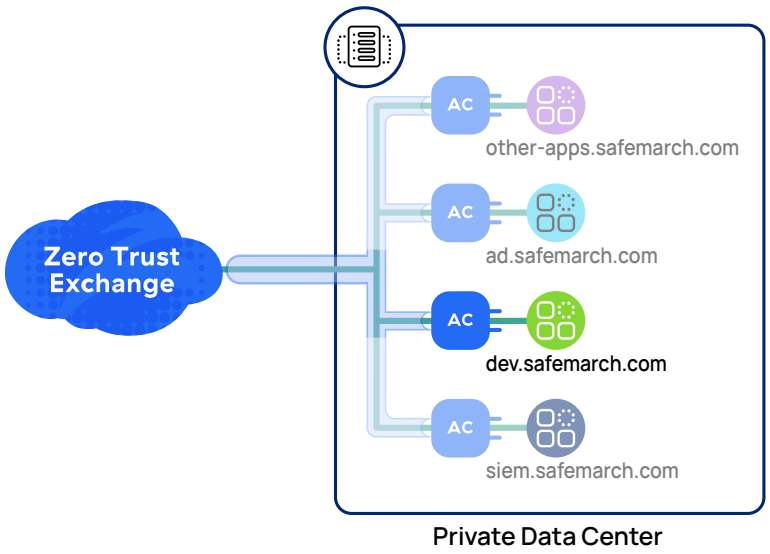


Figure 27. An application only available in the HQ data center

In the preceding image, the "dev" application is only accessible from the HQ data center. It receives its own App Connector group, as these are the only App Connectors that can reach the application.

Server Groups

The primary function of a server group is to associate an App Connector group with an app segment.

The best practice is to create a minimal number of server groups: either one server group per App Connector group, or one server group for each set of App Connector groups (primary and backup) that serve a specific back-end app segment.

The only reasons to create multiple server groups are to control which connectors are queried for a specific application and to minimize the number of health checks. If you know an app is only reachable by certain App Connector groups, you should create a server group that is associated with those App Connector groups and assign it to that app.

Server groups should have Dynamic Server Discovery enabled by default for almost all use cases; only in a few corner cases is it helpful to define servers manually.

Application Access

Policy changes need to be made to refine which users can see which applications. Post discovery, you'll have a better view of what's happening, so you can start to refine the policy for users. How you group users will be organizationally specific, but you likely already have groups or business units that need access to the same applications.

Your application access is likely to need modifications to restrict user access to various applications. For example, let's say you have two applications: you run an application called *finance.safemarch.com* for your finance team, and one for your dev team at *dev.safemarch.com*.

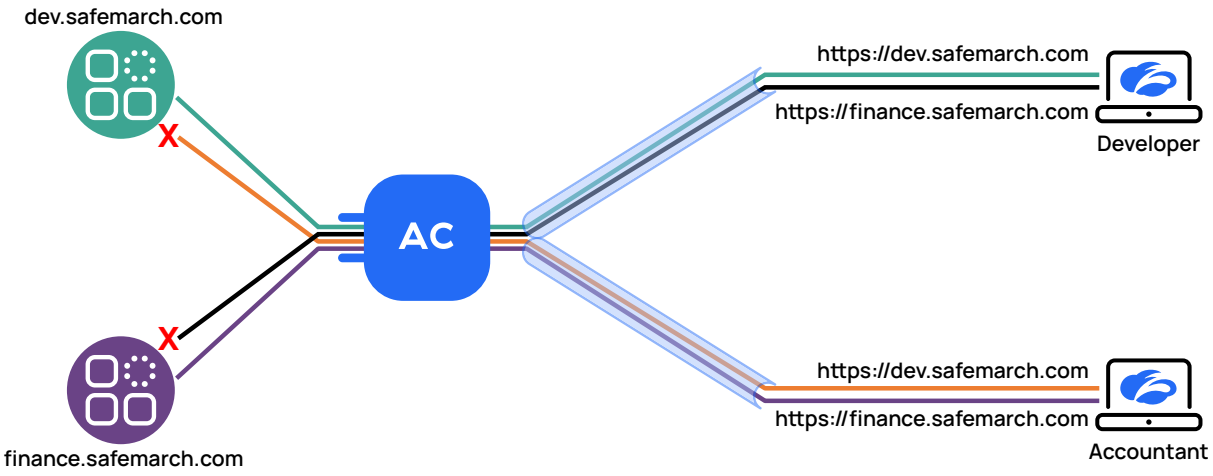


Figure 28. Access control is handled by the application

In the initial policy step, you configured for discovery. Both teams can see the application. Everything was allowed and you relied on the application's existing security to prevent unauthorized action, just as it did before ZPA.

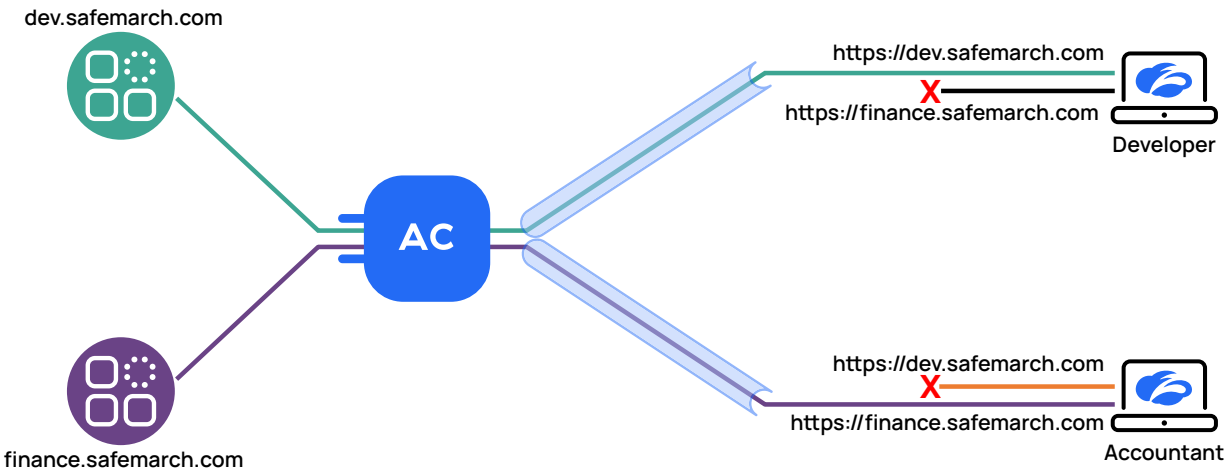
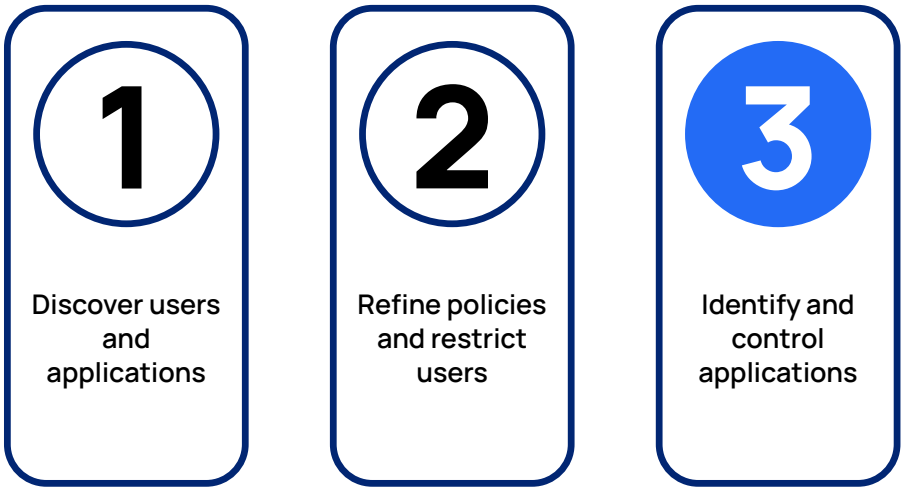


Figure 29. Policy adjustments prevent users from discovering applications

Now that you have more information, there is no reason to allow teams that don't need access to an application to even reach its server. Instead, you can adjust application segments to host those specific applications, and you can then apply enhanced policy to limit access to the appropriate users.

Phase 3 – Identify and Control Applications



At Phase 3, your ZPA service is running, and you have started to control access to critical applications. Now you can start to further refine applications. In this section, we look at additional controls that can be put in place.

Browser-Based Access

The ability to access applications via a web browser, without installing Zscaler Client Connector, is a flexible option for user access. Browser access can provide secure access to HTTP and HTTPS applications, and optional browser isolation provides additional data protection.

Typically, you would use browser access when you can't install an agent, or when you need an additional buffer between the application and the end user. Examples can include:

- Control for user access from devices that do not support Zscaler Client Connector.
- Outside contractors that need to access building systems and only those systems.
- Third-party manufacturers that need to update build records or receive information from you to complete their work.
- The ability to prevent users from directly accessing applications by rendering the application in their browser, as opposed to opening the applications locally.

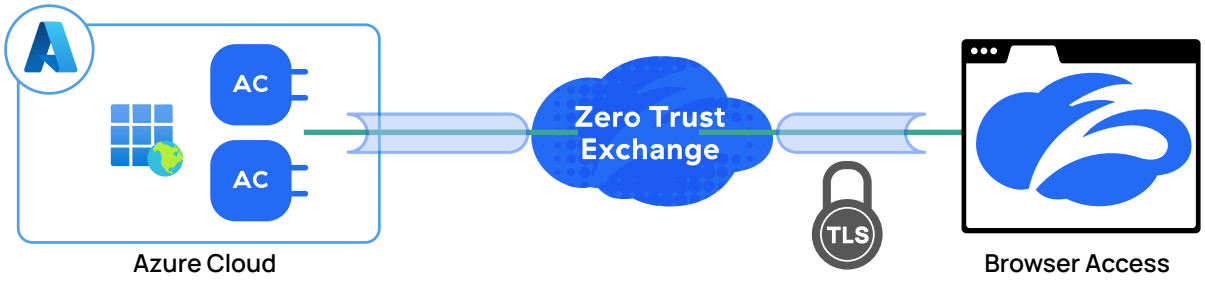


Figure 30. ZPA browser access

Enabling browser access also automatically enables Zscaler Client Connector access to the same applications.

Zscaler recommends only enabling browser access in cases where needs cannot be met with Zscaler Client Connector, or where access to systems is restricted to parties outside the organization.

To learn more, see [About Browser Access](https://help.zscaler.com/zpa/about-browser-access) (https://help.zscaler.com/zpa/about-browser-access).

Double Encryption

In a standard ZPA connection, all traffic flows through a TLS 1.2 tunnel. For most traffic, that is sufficient, as your applications are likely to be using end-to-end encryption (HTTPS, SSH, RDP, etc.). In these cases, you already have double encryption by default, as the application tunnel is inside the ZPA tunnel. When stored in system memory, it's stored in its original encrypted state.

However, should you need to support legacy applications that do not have end-to-end encryption (HTTP, FTP, Telnet), double encryption allows you to add a second layer of encryption to the transmitted data.

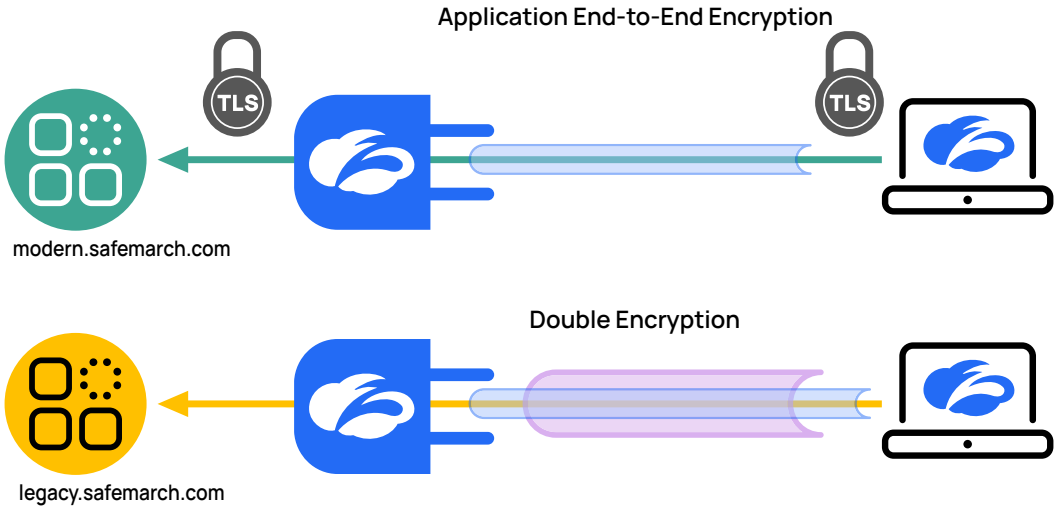


Figure 31. Application end-to-end encryption vs. double encryption by Zscaler Client Connector

Double encryption requires some additional certificate work. The Zscaler App Connector and Zscaler Client Connector certificates must share the same root central authority (CA). This CA can either be the default CA created when ZPA is provisioned, or it can be your own public key infrastructure (PKI) CA. If you want to ensure that traffic data is never accessible within the Zscaler cloud, even when transiting the ZPA Service Edge, you must use your own PKI.

It is important to understand that double encryption is applied at the domain level. All apps sharing the same domain name, such as *finance.safemarch.com*, no matter the port definition, have double encryption enforced. Before enabling double encryption on an app segment, be sure to find out what other app segments are also using the application name.

Double encryption requires additional processing by the Zscaler App Connector. This processing overhead impacts the throughput of the connector. If your throughput falls, you need to launch additional connectors. Zscaler strongly recommends setting your connectors to autoscale to meet demand in a double encryption deployment. To learn more, see Understanding App Connector Throughput (https://help.zscaler.com/zpa/connector-deployment-prerequisites#ConnectorThroughput).

To learn more about double encryption, see About Double Encryption (https://help.zscaler.com/zpa/about-double-encryption).

Health Reporting

When you configure your application segments, you can also enable health monitoring for the applications. When you set up health monitoring, the App Connector attempts to connect to your applications using the information in the app segment definition.

The App Connector uses local DNS resolution to resolve FQDNs to IP addresses; each IP and port combination is called an application target. The combinations are as follows:

- FQDN – Check all IP addresses returned via DNS resolution.
- IP address – Check the addresses in the configuration.
- TCP access – Perform a TCP three-way handshake to test reachability.
- UDP access – Perform an ICMP check and, in the case of failure, try a TCP check.

The App Connector, with a list of IPs and ports, iterates through all of them. If an FQDN and an IP are both provided, all the IPs are merged into the list. Each TCP port and each UDP port is tested on every listed address.

For example, your application *jira.safemarch.com* returns two IP addresses. Its application segment is listed as follows:

```
FQDN: jira.safemarch.com
IP: 10.10.10.10
TCP port: 80
UDP port: 6556
DNS IP response: 10.10.10.12, 10.10.10.14
```

The system potentially performs six health checks, as there are six application targets:

```
10.10.10.10  TCP 80
10.10.10.10  UDP 6556
10.10.10.12  TCP 80
10.10.10.12  UDP 6556
10.10.10.14  TCP 80
10.10.10.14  UDP 6556
```

Zscaler recommends that you initially enable health reporting on your mission-critical apps. These are the same apps that you restricted access to in Phase 2. The best practice is to use on-access health checks—which disables background health checking when an application is not in active use—unless you explicitly desire the health dashboard to always show the current health status of an app and never show the app health in an "Unknown" state.

For other applications, you'll want to take into consideration the number of health checks needing to be performed versus the speed at which the data can be fed back into the platform.

The App Connector is limited to 20 checks per second, and a maximum of 6,000 application targets. Polling 6,000 application targets takes approximately 5 minutes. As with all monitoring, you should strive to keep the polling interval as short as reasonable for the best performance.

If your configuration results in more than 6,000 health check targets, all further targets are ignored.

Bypass Settings

When an application is better served outside of ZPA, you can use bypass settings. In these cases, the settings that would otherwise be applied are overruled and the traffic is handled normally outside of ZPA. There are two primary scenarios where bypasses are configured:

1. Bypassing ZPA when on the corporate network
2. Bypassing ZPA for particular applications, regardless of network

In the first use case, the goal is to have applications served from the local data center when on the organization's network. You need to configure a forwarding profile to define the corporate network. When the Zscaler client detects that it is operating from the corporate network, the client bypasses ZPA. This setting is only appropriate for apps hosted in Azure if you desire the user to follow the local network path to the application (e.g., from the user's location to a data center that has ExpressRoute to the Azure environment), rather than leveraging ZPA's dynamic path optimization to deliver traffic to the application.

The second case deals with applications that might appear due to auto-discovery, but do not need—or would be inappropriate—to be accessed via ZPA. The easiest example is that of the corporate website. It's a publicly available site, but if you have split DNS (same external and internal DNS domain), it will be found via the wildcard selection for the domain, and ZPA still attempts to handle it.

The use of bypass settings is dependent on organizational policy and application. Therefore, Zscaler makes no general recommendation about the use of bypass settings.

Leveraging the ZPA API

For teams looking to expand their automation capabilities with ZPA, Zscaler offers a robust API interface. This interface allows you to dynamically update configuration and policy from automation and orchestration systems. With the ZPA API, the following components can be changed programmatically:

- Application Segments
- Segment Groups
- Servers
- Server Groups
- App Connectors
- App Connector Groups
- Browser Access Certificates
- Cloud Connector Groups
- Customers
- Enrollment Certificates
- IdP Configurations
- Access Policies
- Client Forwarding Policies
- Timeout Policies
- Log Streaming Service (LSS) Configurations
- Machine Groups
- Posture Profiles
- Provisioning Keys
- SAML Attributes
- SCIM Attributes
- SCIM Groups
- Service Edges
- Service Edge Groups
- Trusted Networks
- Version Profiles

A deep dive on the Zscaler API is beyond the scope of this guide. If your development or operations team wants to leverage the ZPA API, Zscaler provides an **API Developer & Reference Guide** (**https://help.zscaler.com/zpa/zpa-api/api-developer-reference-guide**).

About Zscaler

Zscaler (NASDAQ: ZS) accelerates digital transformation so customers can be more agile, efficient, resilient, and secure. The Zscaler Zero Trust Exchange protects thousands of customers from cyberattacks and data loss by securely connecting users, devices, and applications in any location. Distributed across more than 150 data centers globally, the SASE-based Zero Trust Exchange is the world's largest inline cloud security platform.