Zero Trust Access to Private Apps in AWS with Zscaler Private Access™

Reference Architecture

Table of 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 of the guides will steer you through the architecture process. They will provide technical deep dives into specific platform functionality and integrations.

The Zscaler Reference Architecture series is designed to be modular. Each guide will show you how to configure a different aspect of the platform. This allows you to use just the guides 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 will provide a brief refresher on the platform features and integrations being covered. A summary of the design will follow along with a consolidated summary of recommendations.

The rest of the document is written with a technical reader in mind. It will include detailed information on the recommendations and the architecture process. For configuration steps, we will link 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

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

– Warnings indicate that a configuration could be risky. You should read the warnings carefully and exercise caution when making your configuration changes.

Finding out more

You can find our guides at **https://www.zscaler.com/resources/reference-architecture**

You can join our user and partner community and get answers to your questions at **https://community.zscaler.com**

Icons used in this guide

The following icons will be used in the diagrams contained in this guide.


Zscaler Cloud

AWS Cloud

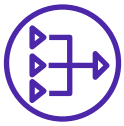

Private Subnet


App Connector


Internet Gateway


Public Subnet

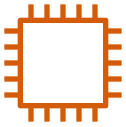

Laptop with Client Connector


NAT Gateway


Auto Scaling Group


Phone with Client Connector


Router


Instance

TLS Tunnel


Amazon Direct Connect


Data Center


VPN Gateway


VPN Concentrator

AWS Cloud


Internet


Region


Router


VPC

Introduction

Zscaler Private Access (ZPA™) is a part of the Zscaler Zero Trust Exchange™ platform. ZPA provides secure access, based on a zero-trust framework, to private applications — including remote access to internal applications running on Amazon Web Services (AWS). 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 AWS that supports any device and any internal application.

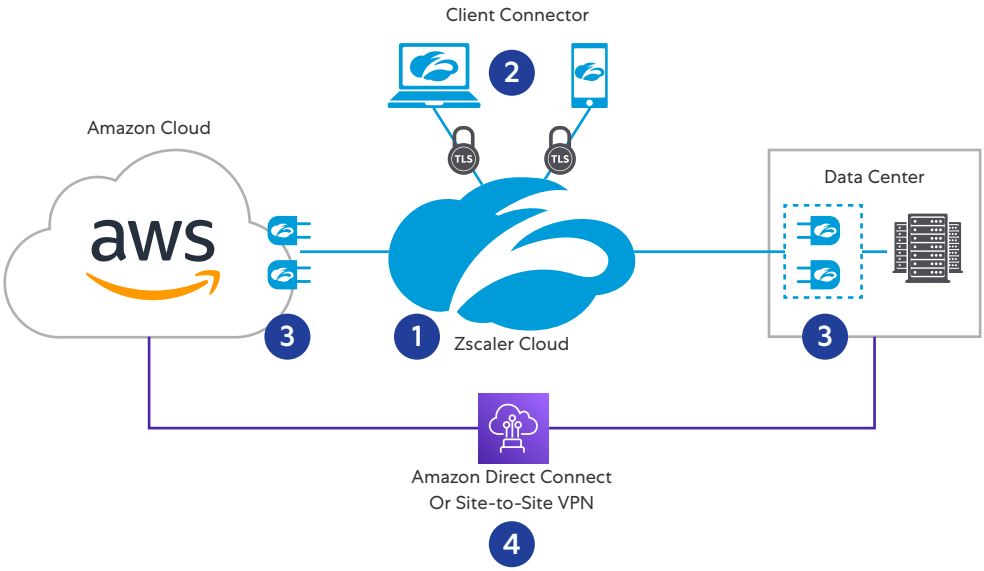


Figure 1: Overview of ZPA deployed in AWS

The image above 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 AWS 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. **AWS Direct Connect or site-to-site VPN (Optional) –** As you transition to the cloud, you likely have applications in your on-premises data center that are also in AWS. It may also be the case that you will continue running applications or databases in your data center that are accessed via AWS front-end applications. AWS Direct Connect is used to synchronize those services via a leased line. You can also set up site-to-site VPNs between your organization and AWS.

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

Wondering about ZPA and zero trust?

If this is your first time reading about ZPA, we encourage you to watch this three-minute overview. In this video we'll cover the benefits of ZPA over traditional VPN solutions: **https://youtu.be/1KLbE243dLY**

Need to share a quick overview of ZPA and AWS? Please visit our blog published with AWS at: **https://aws.amazon.com/blogs/apn/how-to-securely-access-amazon-virtual-private-clouds-using-zscaler-private-access/**

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

Interested in learning more about zero trust? Visit our zero trust microsite at: **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 the 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 from providing network access to allow access to applications. Instead, users receive specific access based on authentication results.

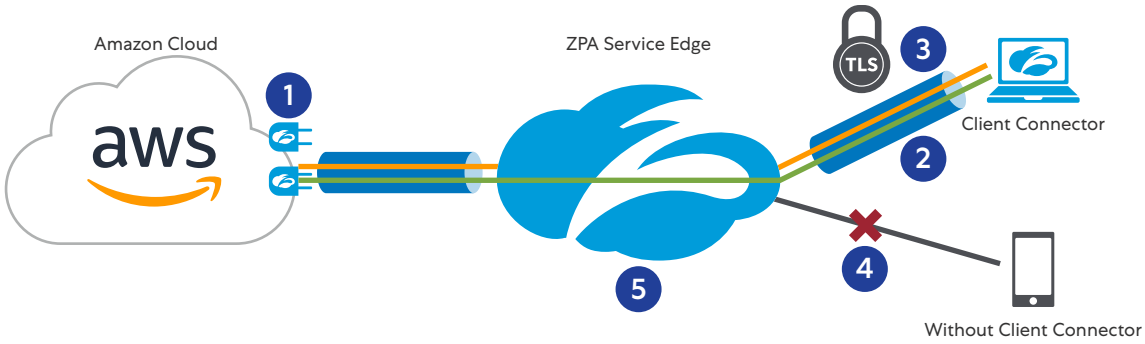


Figure 2: Application access with ZPA

In the diagram above, one of the users has two devices: a company-issued laptop with Zscaler Client Connector, and a phone that does not have Client Connector installed. When the user on a company-issued laptop attempts to connect to an internal application hosted in AWS, 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 AWS 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 VPC (yellow path).
2. The user's laptop running Zscaler Client Connector establishes its own independent data-plane microtunnel to the nearest ZPA Service Edge. 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 (orange path). The request is intercepted by the Zscaler Client Connector, and if it is marked as an internal application, it is forwarded to the ZPA Service Edge for a policy decision. If the application is not an internal app, the traffic resolves normally outside of the ZPA service.
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 will resolve, and the connection will begin. If the user is blocked by policy, the application will not resolve; in effect, it will not exist for that user. No matter the outcome, all policy requests are logged by the Zscaler system.
5. The ZPA Service Edge signals both the Zscaler Client Connector and the appropriate App Connector to begin building a microtunnel that will be carried over the existing data-plane tunnel (green path). This new microtunnel exists only for application traffic between a user and an application. Each application will have 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 travel app via the company-issued laptop will not be able to connect using their phone; they will not be able to resolve the application's existence. Because it is only resolvable via internal DNS, and the phone does not have Client Connector installed, the application is invisible to the user. The application doesn't have a public IP, so there is no way for the user to browse to the application directly. The only access option is the App Connector.

Even if an internal user knows an app exists, the user will not necessarily be able to 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.

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 just that application for just that authorized user.

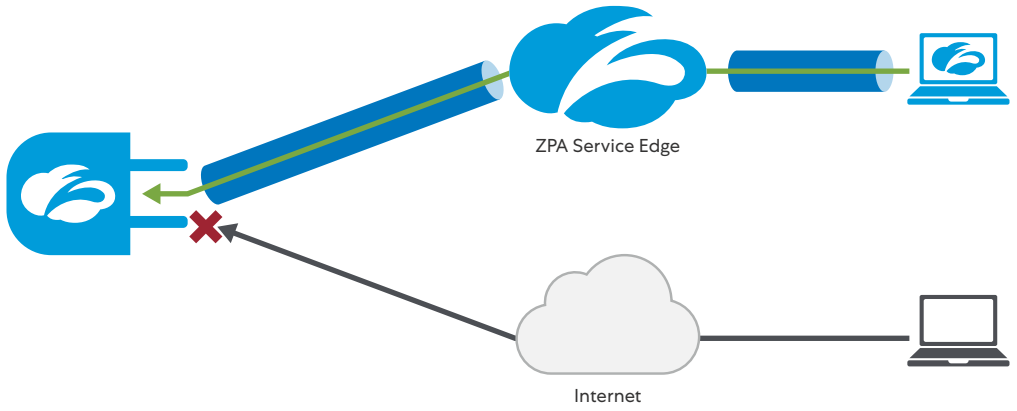


Figure 3: App Connector and Client Connector each create outbound-only data-plane tunnels

ZPA also protects your private applications by allowing you to make them invisible to the internet. Since these applications will 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, multifactor 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.

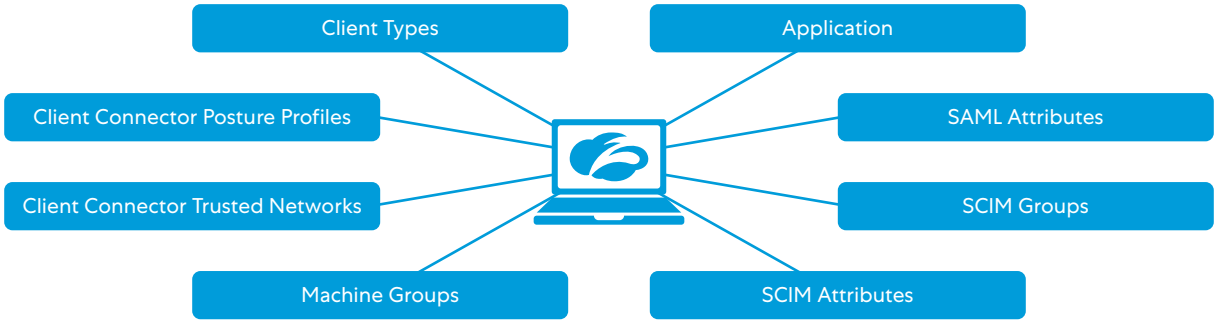


Figure 4: ZPA policy inputs

Differentiating policy on more than a user's role in the organization can help mitigate data leakage or infection. As an 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 may wish to restrict the user's views or deny access altogether.

Deploying ZPA in AWS

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

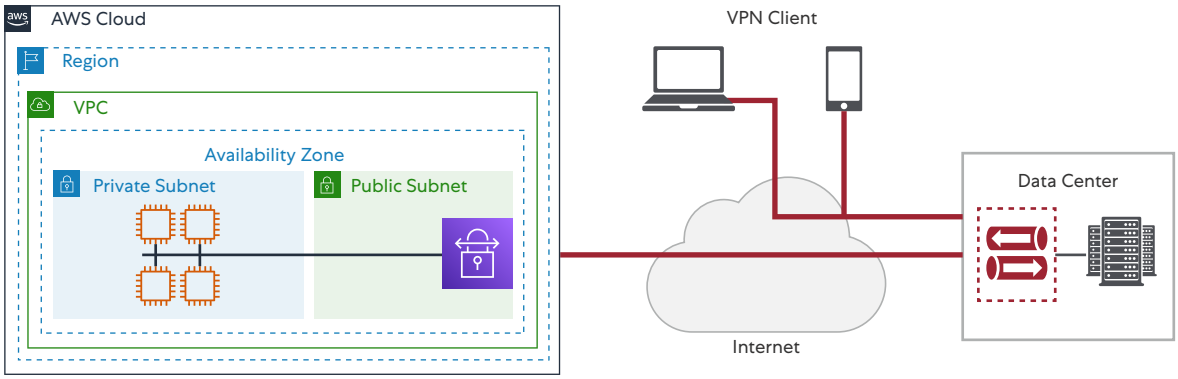


Figure 5: AWS access via legacy VPN

Prior to ZPA, users would likely have taken one of a few routes to access AWS instances. In most deployments, a VPN has been used to provide access to internal applications, which are not exposed to general internet access. You might also pay to have an AWS Direct Connect line back to your data center. 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.

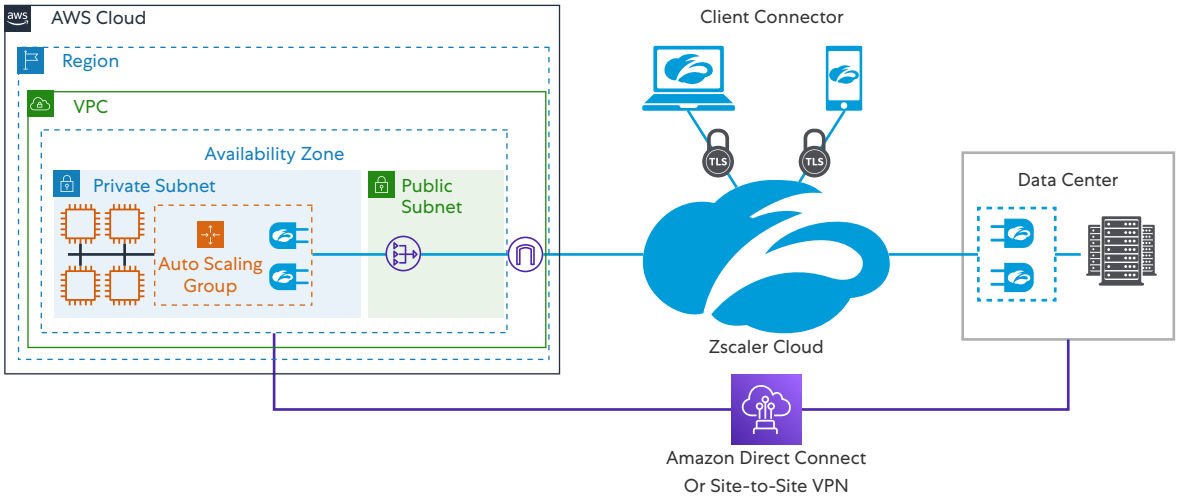


Figure 6: AWS access with ZPA

The ZPA model shifts away from backhauling traffic and using a VPN that puts users on the network. The primary changes will be the removal of your VPN gateways, as they will no longer be needed for access.

Instead, you'll deploy groups of two or more Zscaler App Connectors to each of your VPCs with internet access. This could be the same VPC that contains your applications, or to a gateway VPC serving private VPCs without direct internet access. These will be 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 will allow you to deploy them on reserved instances.

Your users will run the Zscaler Client Connector (https://help.zscaler.com/z-app/what-zscaler-app), 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 please visit:

- ZIA: **https://www.zscaler.com/products/zscaler-internet-access**
- ZDX: **https://www.zscaler.com/products/zscaler-digital-experience**

AWS 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 will not be aware of whether they are accessing AWS or the corporate data center. ZPA simply steers the client to the nearest App Connector that can provide application access.

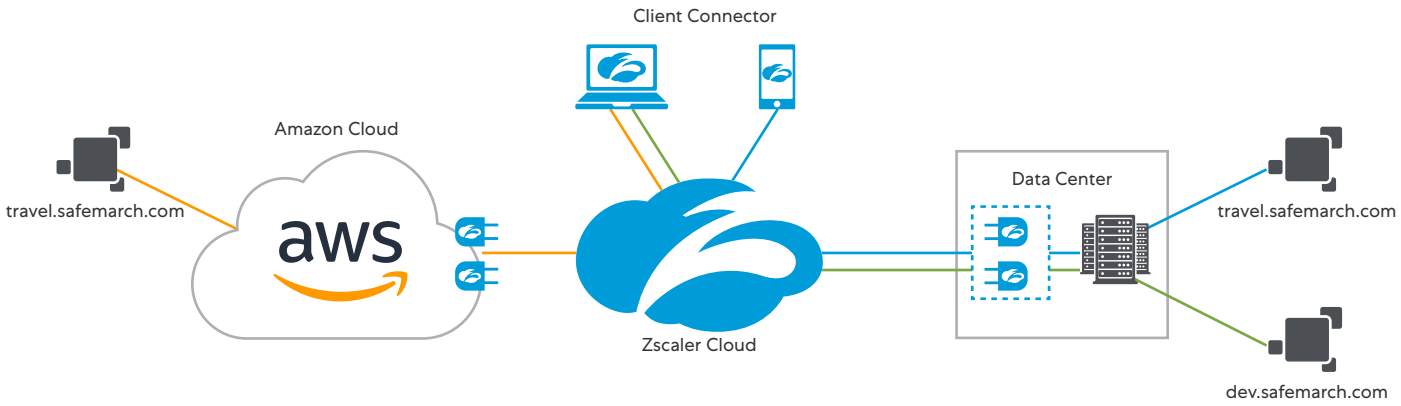


Figure 7: Hybrid access allows apps to exist in multiple places or in your data center only

Amazon Direct Connect is a service from AWS that links your on-premises data center with the AWS cloud. This service allows your applications to remain synchronized between AWS and your data center. Client Connector on your users' devices will be directed to the best location to connect to the applications. You can also run applications seamlessly that only exist in AWS or your data center.

Adding ZPA to your AWS deployment

The process for connecting ZPA and AWS requires deploying virtual machines into your AWS VPCs provisioned with access to your applications and the ZPA Service Edges. The App Connectors will then reach out to the Zscaler cloud to establish a connection and receive configuration. Once up to date, the App Connectors will be available to service your user traffic to applications.

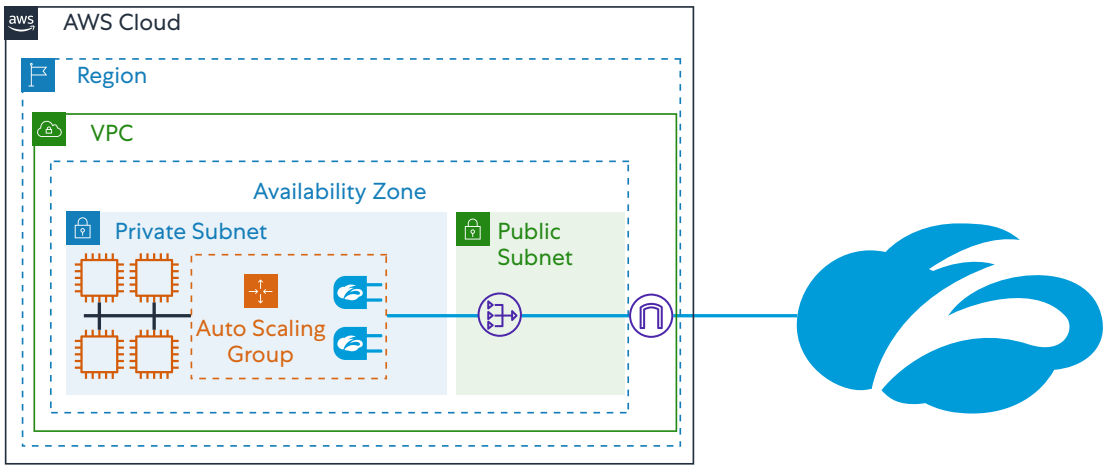


Figure 8: Add Zscaler App Connectors to your VPC

App Connectors

Zscaler publishes the App Connector software in a variety of formats for various platforms. On AWS, Zscaler offers an App Connector Amazon Machine Image (AMI), 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 AMI disables all unnecessary listening services to further reduce your attack surface. This document focuses on App Connector deployment via the AMI.

The App Connectors are initially provisioned in the ZPA admin panel 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 will configure per-VPC 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 geo-location of the connectors and the apps the connectors can serve, and they are used by Zscaler to manage regular upgrades.
- **Application discovery –** ZPA will initially help you identify all the applications you currently have running in AWS by observing user traffic. The system will report on what applications it serves and which users are accessing them.
- **Policy definition –** Once 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 AWS side, we recommend using AWS Launch Templates to consistently spin up App Connectors. These will include:

- The approved Amazon Elastic Cloud Compute (EC2) instance that can support the App Connector AMI.
- The provisioning key for the App Connectors to contact Zscaler.
- The configuration of AWS Auto Scale groups to ensure consistent capacity to the App Connectors.

App Connector connectivity

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

The App Connectors themselves will 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 AWS NAT gateway.

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. This follows the AWS shared responsibility model. 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 will upgrade 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 your App Connectors with all the connectors in a group in a single geographic location. You can then schedule that group to be upgraded during low-use periods in that time zone.

Defining policy

Zscaler follows a three-phased approach to defining policy.

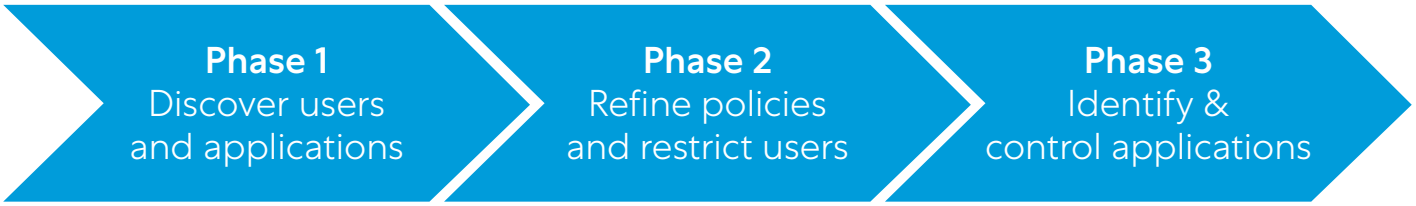


Figure 9: Policy definition framework

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.

As an example, your engineers probably don't need access to your company financials or the accountants to your development platforms. These are generally straightforward applications with limited numbers of users.

Phase 3 – Identify and control applications

At this point, you will 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 may 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 will 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 as they do, and who specifically needs access to them.
- **Remove or restrict the discovery process –** At this stage, most (or all) of your users will be on board and you won't need to continue discovering applications in the same way. You might remove or disable the discovery policy altogether. Or you may 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 will dive into more detail on each of these topics, providing Zscaler best practices and recommendations for deployment. Each section will link to the appropriate Zscaler Help documents. These documents will show you how to configure the Zscaler service with interface walkthroughs and short overview videos.

Deployments in AWS

Zscaler App Connectors are virtual machines that run in multiple clouds. Deploying Zscaler App Connectors in AWS is similar to deploying other application instances. An existing Amazon Machine Image (AMI) is available and is updated regularly by Zscaler. The following section will cover core concepts about App Connectors to help you understand and plan your deployment. At the end of this section, you should understand:

- Supported AWS image types
- App Connector connectivity requirements
- App Connector security
- Deploying App Connectors with AWS launch templates and Auto Scaling

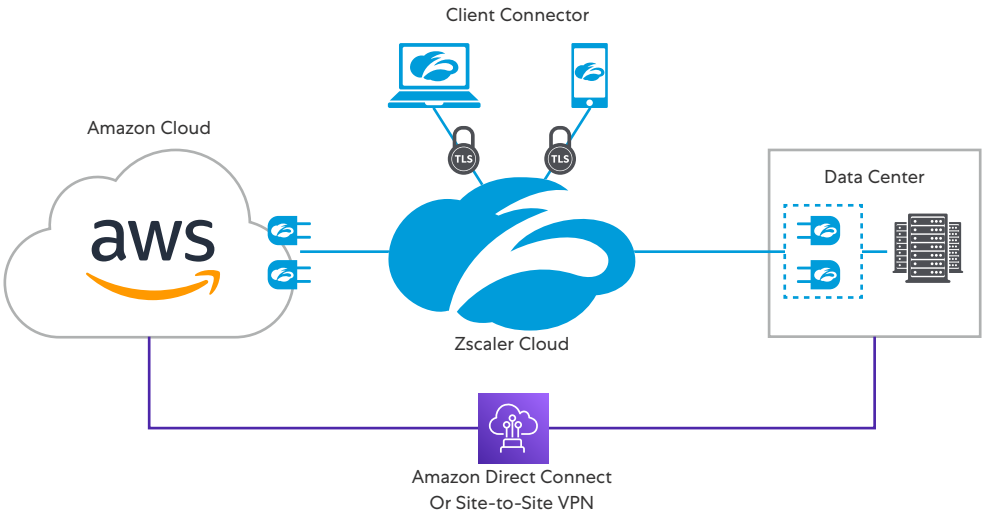


Figure 10: Modified VPC with App Connectors

Supported AWS compute types for App Connectors

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

- Memory: 4 GB RAM Minimum
- CPU: 4 CPU cores (Xeon E5 class) for virtual machines (VMs) with hyperthreading
- Disk Space: 8 GB (thin provisioned) for all the 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 will need to account for this difference when sizing your connectors.

For current, detailed specifications and sizing requirements, please visit: **https://help.zscaler.com/zpa/connector-deployment-prerequisites#SpecsAndSizing**

AWS offers many types of instances in the EC2 environment, but not all of these instances are suitable for production use. Zscaler's best practice for App Connector deployment is to use one of two types:

- **m5a.xlarge –** This should be your choice for all production App Connectors, and it meets all of the requirements.
- **t3.xlarge –** This image should only be used for testing, or applications with very limited needs, as these machines are administratively limited.

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

For more information on the differences and details of the AWS instance types, please visit: **https://aws.amazon.com/ec2/instance-types/**

To learn more about the Amazon Marketplace please visit: **https://aws.amazon.com/marketplace/**

App Connector placement in AWS

Deploying App Connectors in AWS 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 VPC with your application instances
2. Deploying the App Connectors in a gateway VPC that serves other private VPCs

Deploying App Connectors with your application instances

In this model, you'll deploy App Connectors alongside your application instances in the same VPC. The App Connectors exist in the same subnet as the applications they service. This model is shown below:

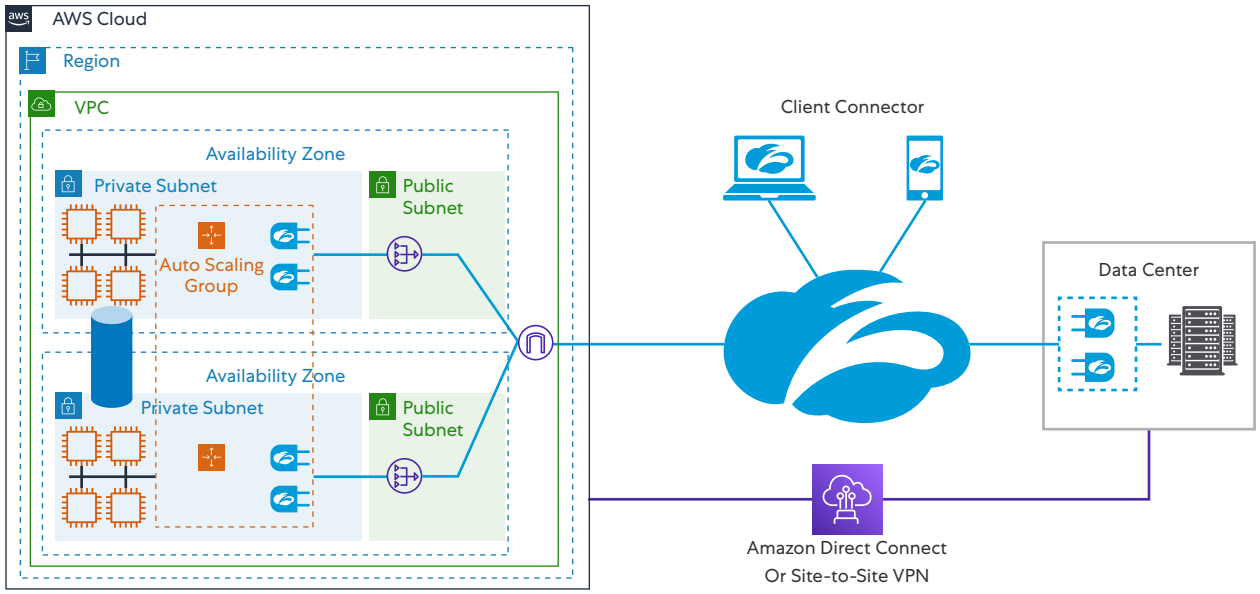


Figure 11: App Connectors in multiple AWS Availability Zones within a region

Gateway VPCs

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

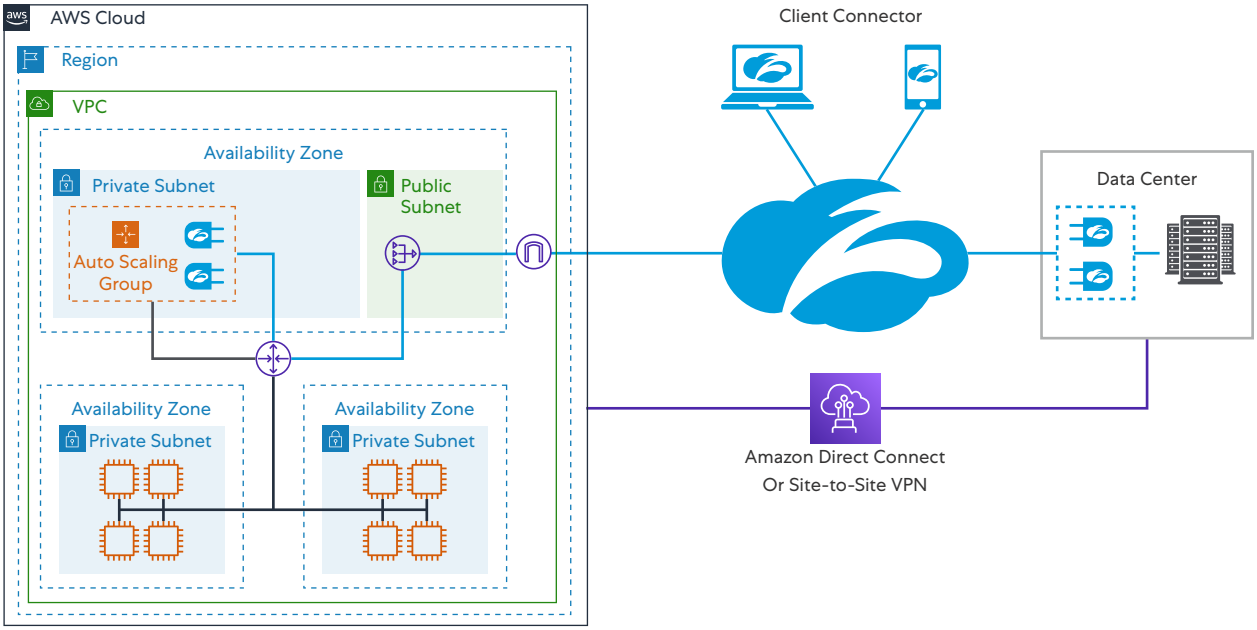


Figure 12: Multiple private VPCs using a common gateway

Availability Zone redundancy

Availability Zones in AWS are Amazon data centers 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 Amazon 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 will fail across to the App Connectors in another Availability Zone.

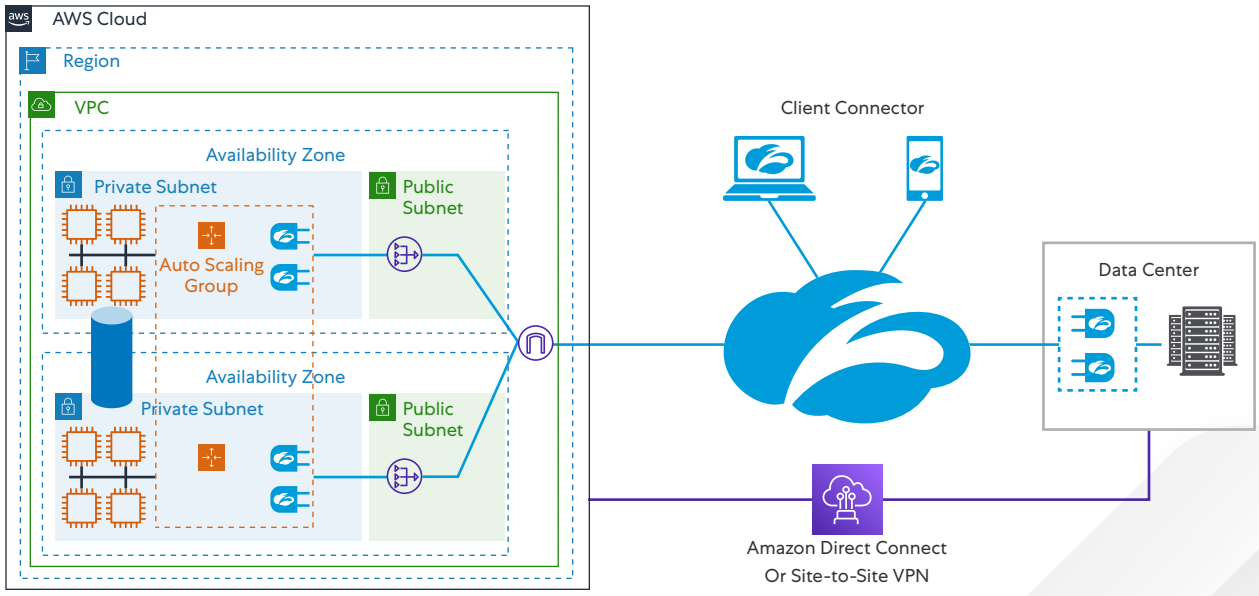


Figure 13: App Connectors in multiple AWS Availability Zones within a region

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, please visit **https://aws.amazon.com/about-aws/global-infrastructure/regions_az/.**

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 an Oregon or Washington data center instead of one in Boston.

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 AWS deployment. You may 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.
    a. 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.

For more details on App Connector connectivity requirements, please visit:
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 AWS recommend using AWS Cloudwatch to monitor access logs. Cloudwatch will provide logs from your Security Groups and Access Control Lists (ACLs). Additionally, check any ACLs on the applications themselves for issues. You may need to make adjustments to account for your connections coming from the App Connectors.

To learn more about Amazon Cloudwatch, please visit: **https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/WhatIsCloudWatchLogs.html**

App Connector network connectivity

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

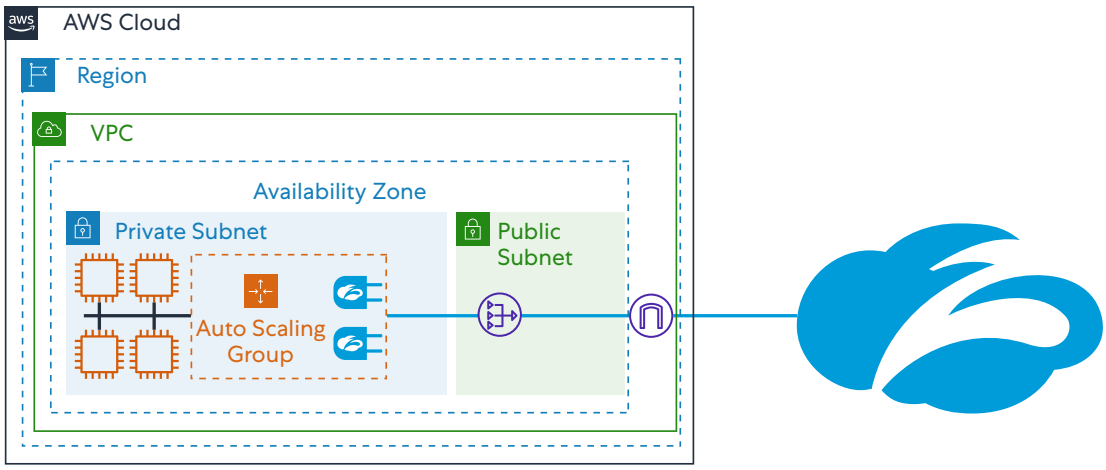


Figure 14: App Connectors behind an internet gateway and NAT gateway

For App Connectors behind virtual network devices, Zscaler's recommendation is to keep App Connectors behind a NAT gateway where they can directly reach your private applications. AWS 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 an IGW and an Elastic IP (EIP) for each App Connector. Like any computing platform, you should take a defense-in-depth strategy to securing App Connectors from direct internet access.

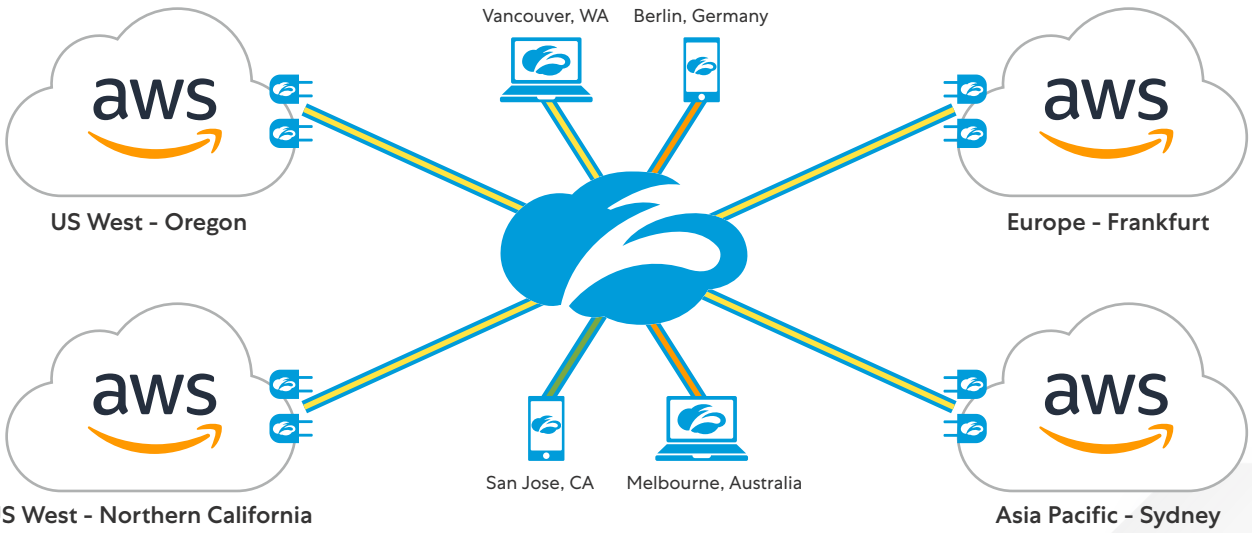


Figure 15: Users will connect to their closest data center

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 be connecting to the nearest ZPA Service Edge. Even when at home, an internet outage by a provider may 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 will provide expanded ranges with 90 days' notice. You can view the ZPA Service Edge IP ranges at: **https://config.zscaler.com/private.zscaler.com/zpa**

For more information on AWS deployment models, see: **https://docs.aws.amazon.com/network-firewall/latest/developerguide/arch-igw-ngw.html**

Application traffic

On the internal side, the App Connector becomes the route for your users to reach your internal applications. The connector and the application need to be able to communicate without being blocked. You will likely need to adjust firewall and ACL rules to ensure any existing security protections are updated. This may 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

ZPA carries client-to-server TCP, UDP, and ICMP traffic between users and applications. These connections are always brokered by the ZPA service to make the connection. This is a model in which the client requests and the server responds. However, once a connection is established, bidirectional communication within the session occurs as usual. Server initiated traffic to clients before a connection is established is not supported.

App Connector software updates

Maintenance of the App Connectors is an automated process handled by Zscaler and is configured at the group level. You will 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 will perform the update process. Once the App Connector is back online, another member of the group will be chosen. This process will continue 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.

To learn how to configure App Connector updates, please visit:
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 on which the App Connector software resides. Please see the section below on maintaining the host OS software.

Host OS software updates

The host OS software will need 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: **https://help.zscaler.com/zpa/upgrading-app-connector-host-os**

There may be cases in which 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: **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 will extend across VPCs as you add more App Connector groups with access to the same application.

Zscaler also recommends spreading your connectors and applications across Availability Zones in AWS to prevent localized outages.

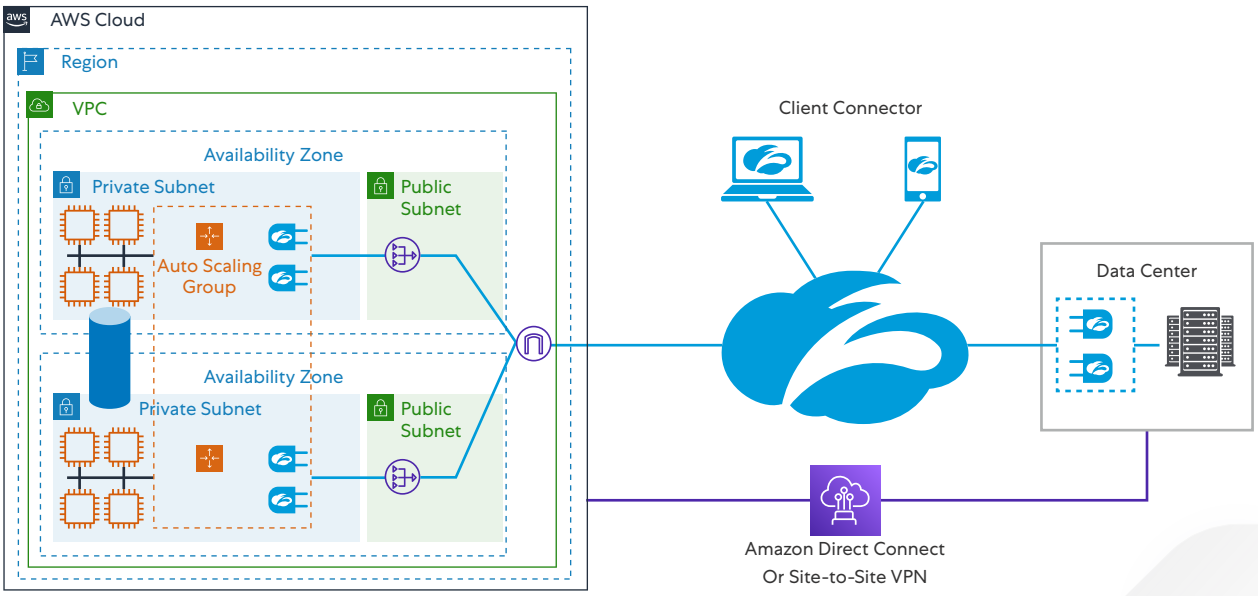


Figure 16: App Connectors in multiple AWS Availability Zones within a region

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 may 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 private cloud (VPC) redundancy

If your application exists in more than one VPC, the App Connector groups for each of the VPCs should be associated with the application (via Application Segment and Server Group mapping - see Applications section below). This allows the ZPA service to connect the user through the nearest data center. If a data center goes offline, connections will automatically fail over to the next closest VPC 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 in to each region, placing connectors from each region in their own connector groups. From there, you will associate all App Connector groups with your application group. For more on planning App Connector groups please see the section: App Connector groups.

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

AWS backend routing and VPC redundancy are beyond the scope of this guide. Please visit AWS help docs for more information at: **https://aws.amazon.com/products/networking/**

To learn more about using Launch Templates please see our deployment guide at:
https://help.zscaler.com/zpa/connector-deployment-guide-amazon-web-services

Deploying App Connectors via launch templates

AWS launch templates allow you to define an instance of an application and quickly deploy a new instance of the application. These are essentially scripted configurations of what you would like AWS to deploy. Zscaler recommends using a launch template for your App Connectors to ensure that the connectors launch consistently each time.

When using a launch template, you tell AWS the size of the instance you need and which Amazon Machine Image (AMI) the template should use. The launch template also allows you to specify components, such as network interfaces, or run configuration scripts.

When setting up your provisioning template, one of the items that can be included is the ZPA App Connector

Some features of launch templates are overridden by Auto Scaling group settings. For more information, see the AWS documentation and the end of the next section.

provisioning key. This key is what ties your App Connector back to your Zscaler tenant. You set the number of times the key can be reused, and you can generate keys for each data center or location. The maximum number of instances tied to a key can be adjusted later if you find you need more App Connectors in a particular location.

To learn more about AWS launch templates, please visit:
https://docs.aws.amazon.com/autoscaling/ec2/userguide/LaunchTemplates.html

Leveraging Auto Scaling groups for redundancy

The AWS service uses the concept of Auto Scaling groups, which allows you to set a minimum, desired, and maximum number of instances of your application. The idea behind these groups is to 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 cost. You will also be ready to expand automatically should something unexpected occur.

Auto Scaling groups are optional, and not required for a ZPA deployment.

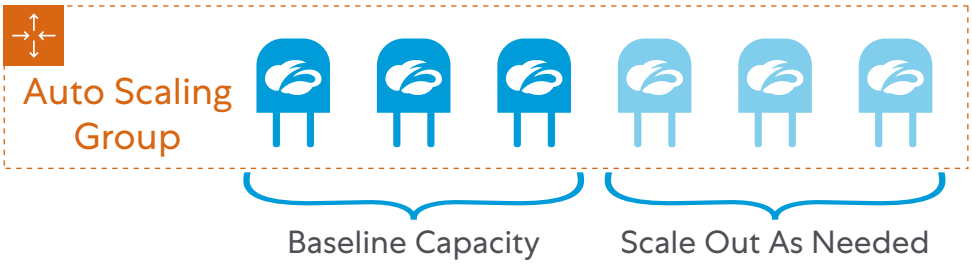


Figure 17: Scale App Connector instances out to meet surprise spikes in demand

When planning your Auto Scaling groups, you should keep in mind that Auto Scaling 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 Auto Scaling groups as a cost-savings method.

Instead, you should think about Auto Scaling groups as a form of redundancy. For example, if you know your users working from your headquarters always hit the same App Connector instances, you will want to scale those to meet your average daily load. Then, you can build an Auto Scaling group to account for any large temporary influxes of users, such as company meetings or customer events you host.

Another reason for building Auto Scaling groups 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 automatically launch a new instance to scale up automatically can help mitigate load issues.

When you no longer need the extra instances, the Auto Scaling group can terminate the instance, but this will likely cause a temporary degradation for users who are connected to that instance. Instead, Zscaler recommends setting up a termination policy within the Auto Scaling group to prevent the new instance from automatically turning itself down. Combined with AWS CloudWatch, you can 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 Auto Scaling and gracefully move users to the remaining App Connectors on reserved instances.

By leveraging launch templates, you can allow AWS 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.

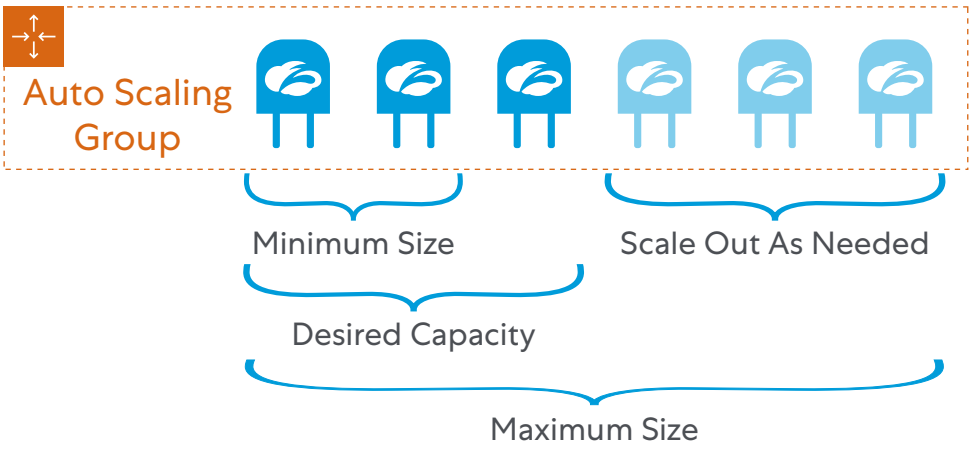


Figure 18: Auto Scaling group of App Connectors

You should set the minimum scale size to match your expected daily load. These would be your reserved 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 Auto Scaling group. If the provisioning key maximum is lower than the Auto Scaling group maximum, new App Connectors will fail to launch successfully.

To learn more about AWS Auto Scaling groups, please visit:
https://docs.aws.amazon.com/autoscaling/ec2/userguide/what-is-amazon-ec2-auto-scaling.html

To view instructions on provisioning launch templates and Auto Scaling groups, including provisioning key scripts, please visit: **https://help.zscaler.com/zpa/connector-deployment-guide-amazon-web-services#Deployment2**

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 may 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 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 workflows you've defined, the user will have an access policy applied. The access policy, as the name implies, defines a user's access to applications.

By leveraging your existing IdP, you don't need to maintain user accounts in multiple places; users will be 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 may 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 some or all of these factors, the user's request is compared against policies with varying levels of access to resources. Access will be determined based on a combination of your policy objectives and the user's complete identity.

Check with your identity and access management team to understand which attributes have been populated by your organization's IdP. You can import your SAML attributes for use by ZPA using the following guide:
https://help.zscaler.com/zpa/about-saml-attributes

Configuration help

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

- On the Zscaler side, configure your IdP. You can find detailed instructions here:
  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 may 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 may be using it.

The need for legacy domain access may 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 the users can access. This is the policy in which 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 may differ depending on how, when, or where the user has requested access. For each policy, the user will be 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.

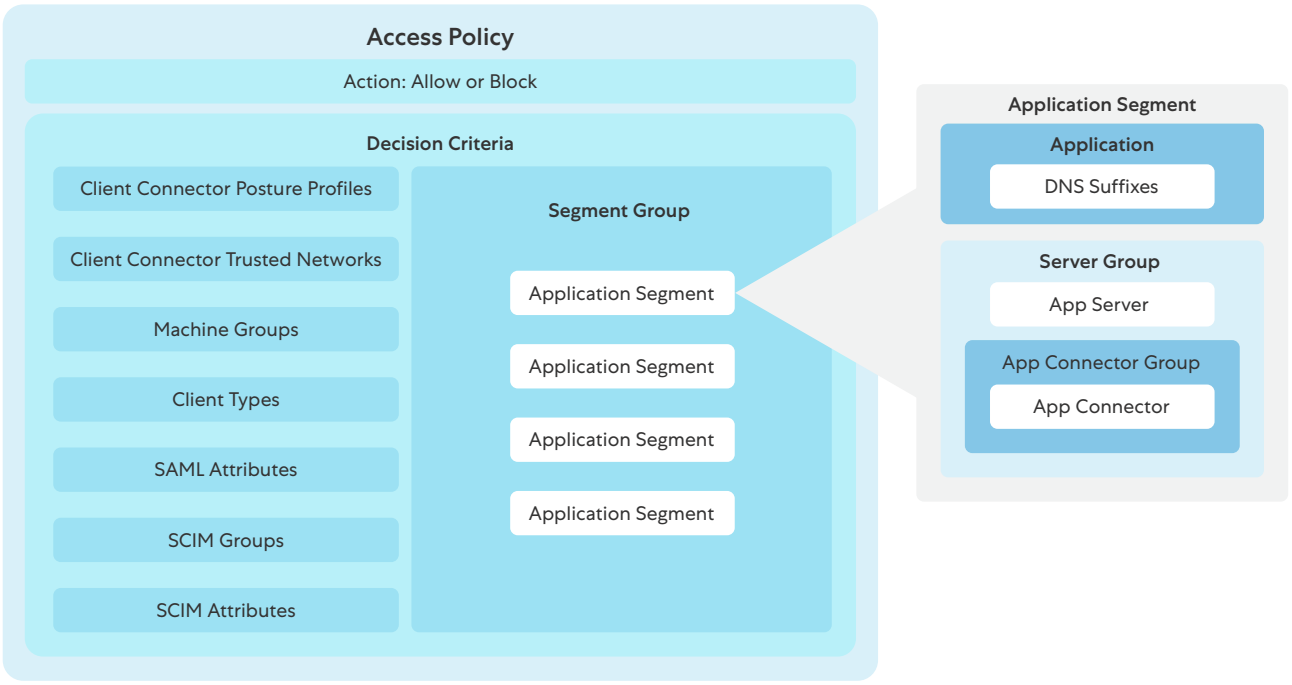


Figure 19: Access policy decision criteria

The following lists include the profiles in the system and how they will be used. We'll 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 in order to access applications with ZPA, observed by the Zscaler Client Connector.
- **Client Connector trusted networks:** Known networks, defined by the administrator, that the Zscaler Client Connector may detect that the endpoint is connected to.
- **Client types:** End-user traffic may originate from Zscaler Client Connector (formerly Zscaler App or Z App), machine tunnel, or web browser. Other traffic may originate from Zscaler Cloud 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 against 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 Private Cloud (VPC) 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 will be 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 AMIs in the AWS Marketplace.

App Connector group – A logical group of deployed App Connectors, with each of the App Connectors existing in a specific group. This group will also be 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 VPC.

Policy definition framework

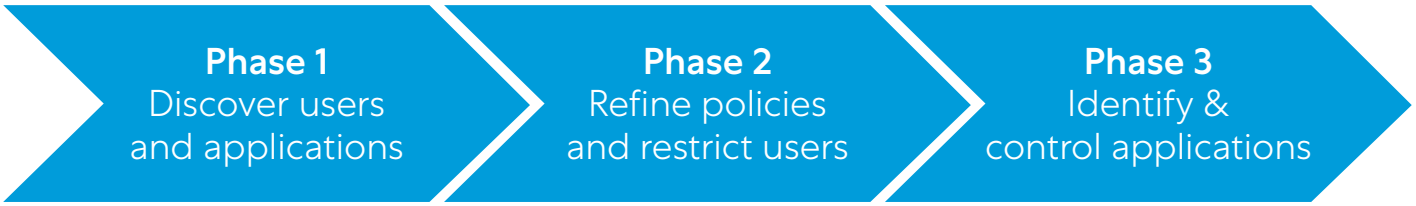


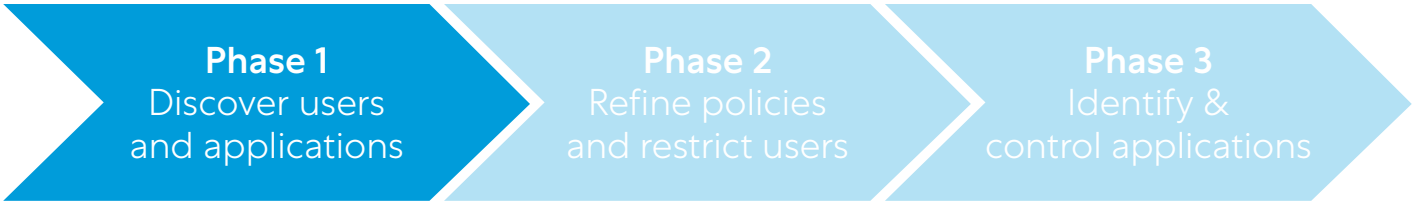
Figure 20: Policy definition framework

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 some additional refinements.

Zscaler recommends this approach instead of simply trying to recreate 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 may 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 may 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 between your users and your applications today. 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 will build in this step will allow all users to see all of the applications where it is 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 will give your team the time to identify and classify apps and users in batches. As more groups are added, you will 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 will be matched rather than a policy for a less-specific target, even if the policy for the less-specific target is placed higher up 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 will be 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 simply 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 will match 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; as such, 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 will do no enforcement. Instead, it will provide 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 could include applications deployed in only a limited number of data centers or an existing physical data center.

To gather this data, you will 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 will gain a solid understanding of applications and services running, how they are being accessed, and by whom. With that application-user interaction map, you will be ready to define more granular policies in the next phase.

For more information on specific dashboards and functions, please see the Zscaler help portal at **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 will help you understand the complete picture of search domains in your organization.

You will 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 the 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 the 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 will want to modernize the application to use an FQDN. This will 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 may 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 still 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.

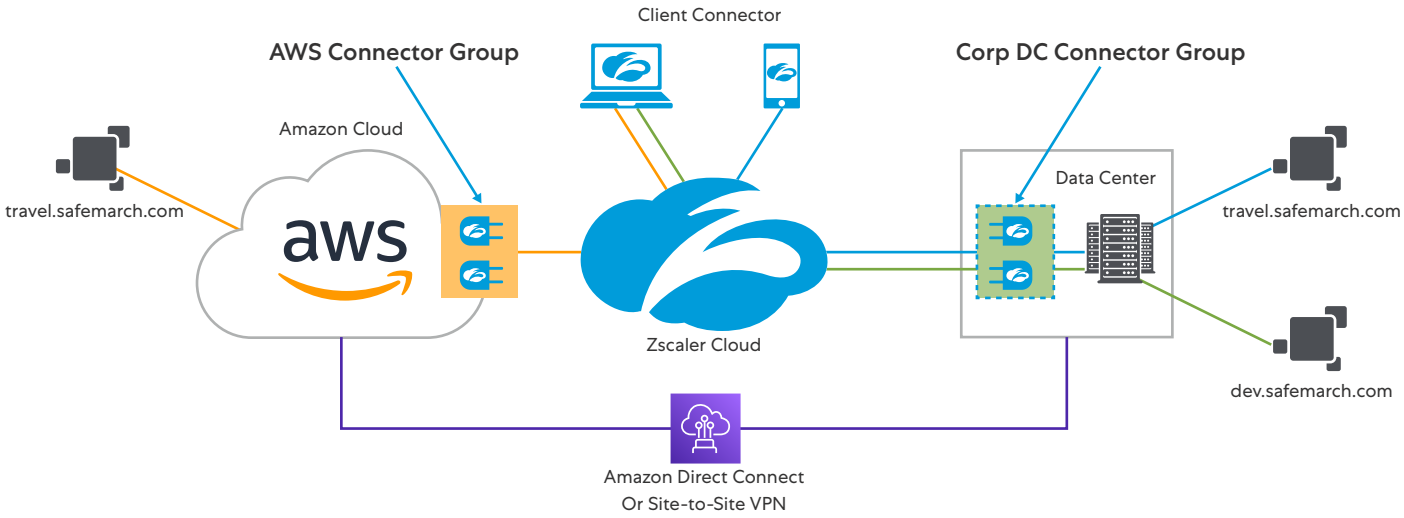


Figure 20: Different App Connector groups have access to different applications

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 show up 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 may 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 will 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 may also need to remove that application from that connector. Some applications may need their own app segment with a modified application list.

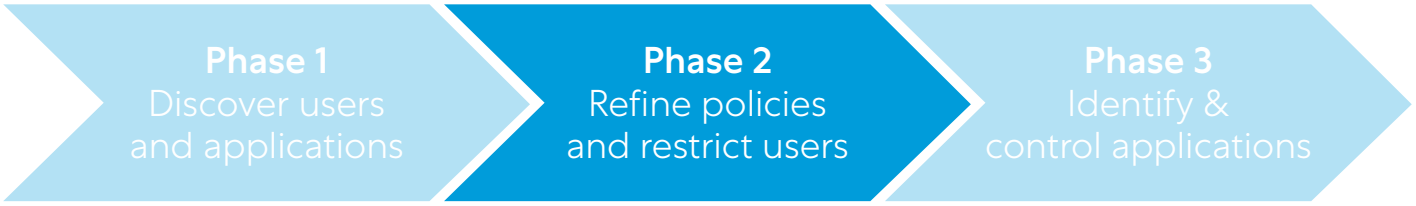
You can drill into any error and get more details about the failure, who tried to access the application, and what functionality it provides. At this stage, you can decide if this is a real problem that needs to be corrected via a policy or configuration change, or if this is simply something that is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

For more information on errors and troubleshooting, please visit: MISSING LINK

User, application, and health dashboards – **https://help.zscaler.com/zpa/dashboard-diagnostics**

ZPA 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 observational data 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 could 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 you will use a combination of the two approaches.

Determine policy by user grouping

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

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

As an example, 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 will often be the approach you take with your more sensitive applications. You'll want to protect those applications and ensure that only privileged users have access.

As you refine your policies, you will become more granular about specific applications. For instance, you may have a web development server that is in AWS 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 will 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.

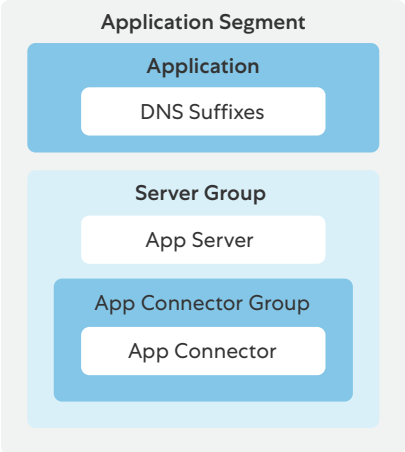


Figure 21: Application segment components

The primary areas of the config you will 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 will make your policy creation that much easier to understand. You will 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 will be able to 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 will 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 will 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 might 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 may 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.

We'll explain App Connector groups based on the simplified diagram above. For this example, we will use a single VPC as a representative example of the rest of the network.

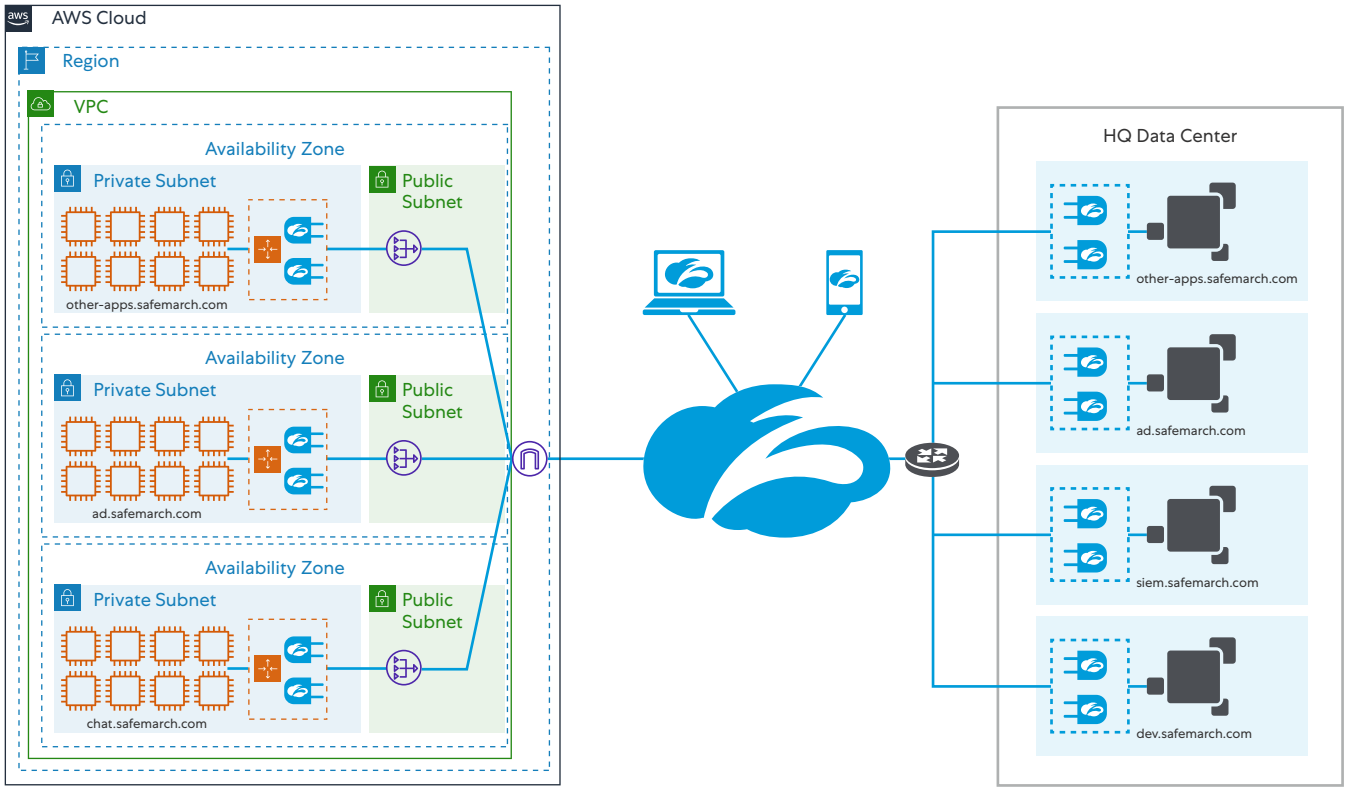


Figure 22: Example network for discussing App Connector groups

Traffic routing

When building out your groups, you should seek to contain the groups to a single AWS region or subset of VPCs 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 will use that information to steer user traffic to the nearest connector.

When a user attempts to connect to an application, the ZPA service will determine 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.

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

Group location-specific applications together in the same App Connector group. You should ensure that your users are being connected to the appropriate, geo-specific App Connectors. If you place all your App Connectors in a single App Connector group, the performance may suffer, as users could 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 will handle multiple applications at a location. There are, however, times when you may 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 anytime 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.

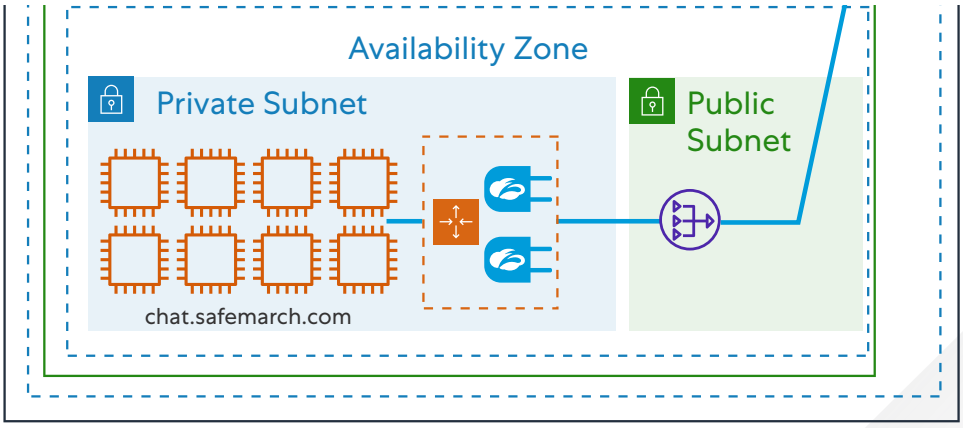


Figure 23: The chat application was impacting other applications and was given dedicated resources

In our example above, the chat application needs a lot of real-time throughput, and in our example 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 24: Active Directory with its own App Connector group

The AD service in AWS 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 will be used. This built-in redundancy means that your AD servers do not need the ZPA service to perform health checks. For more information on health checks, please see **page 43, 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 could be lost. Zscaler recommends at least two App Connectors in the LSS group, and using this group only to service logs streams.

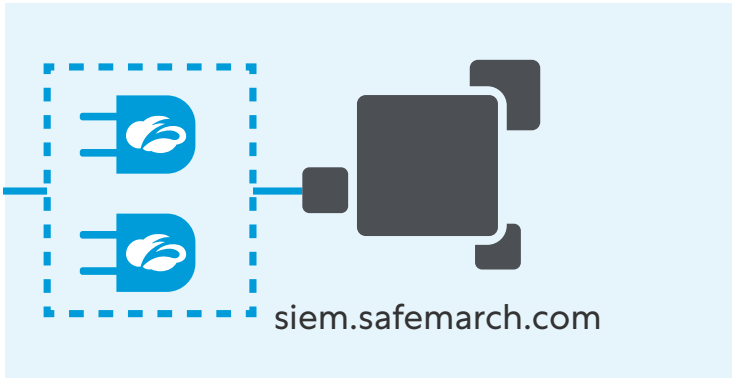


Figure 25: App Connectors for LSS in their own App Connector group

In the example above, 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 may 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 App Connector group.

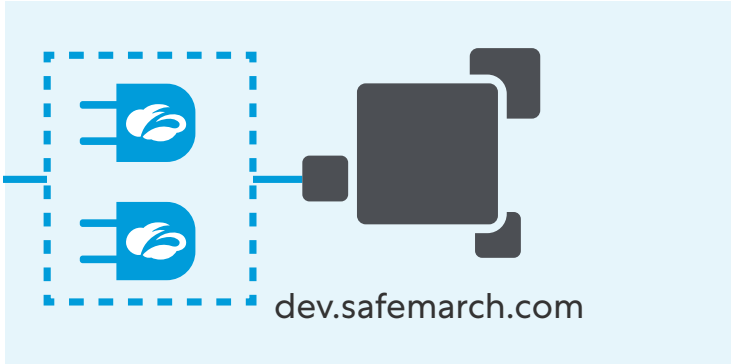


Figure 26: An application only available in the HQ data center

In the example above, 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 DR) that serve a specific back-end app segment.

The only reasons to create multiple server groups are to control which connectors will be 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 will need to be made to refine which users can see which applications. Post discovery, you will 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 generally need access to the same applications.

Your application access is likely to need modifications to restrict user access to various applications. As an 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*.

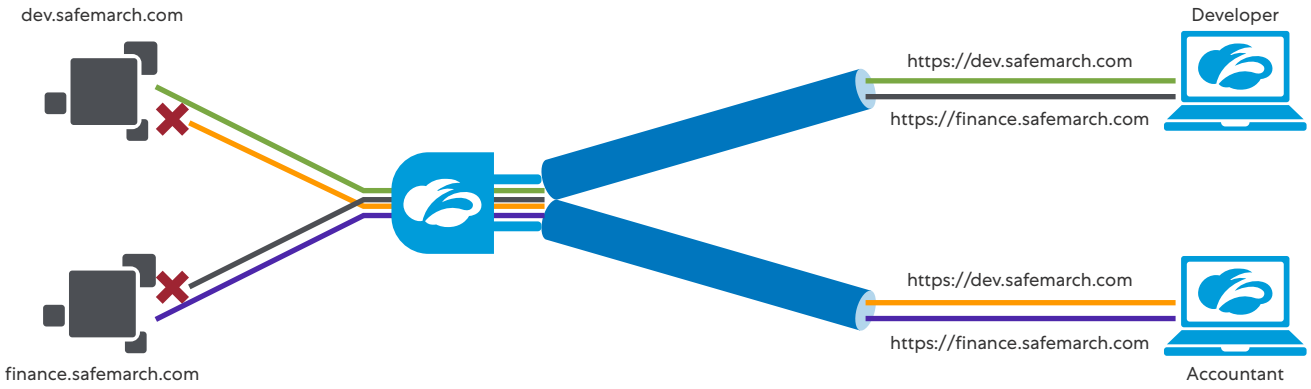


Figure 27: Access control is handled by the application

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.

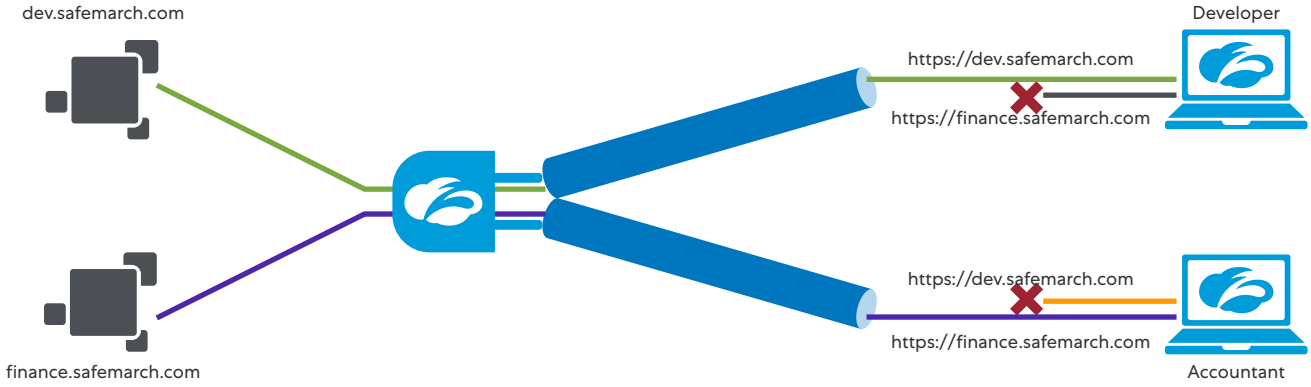
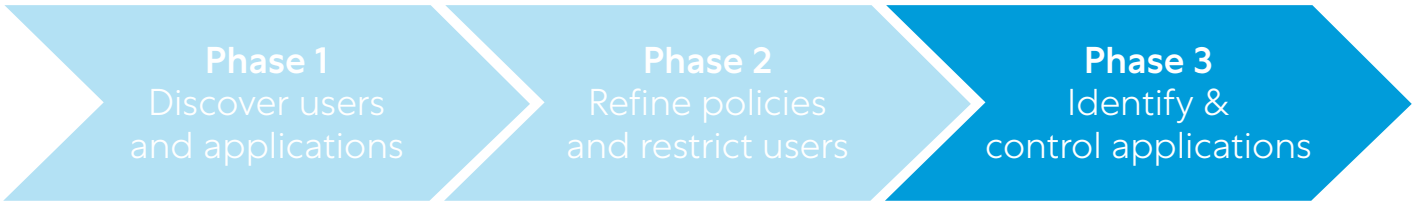


Figure 28: Policy adjustments prevents users from discovering applications

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'll 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 & 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 refine applications further. In this section, we'll look at additional controls that can be put in place.

Browser-based access

The ability to access applications via a web browser, without installing the 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 you need an additional buffer between the application and the end user. Examples may 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

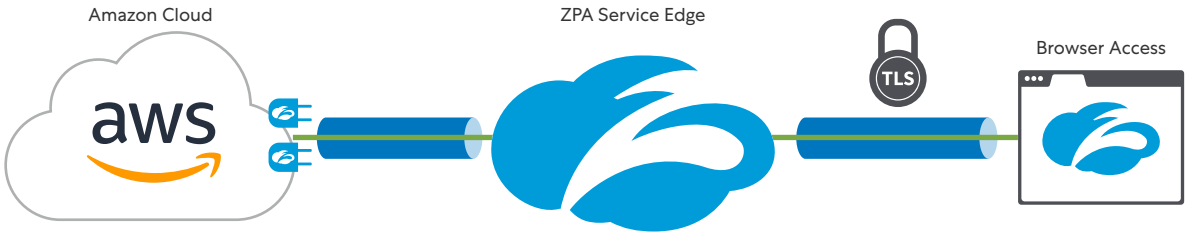


Figure 29: ZPA Browser Access

Note that 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.

For more information about configuring browser-based access, please visit:
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.

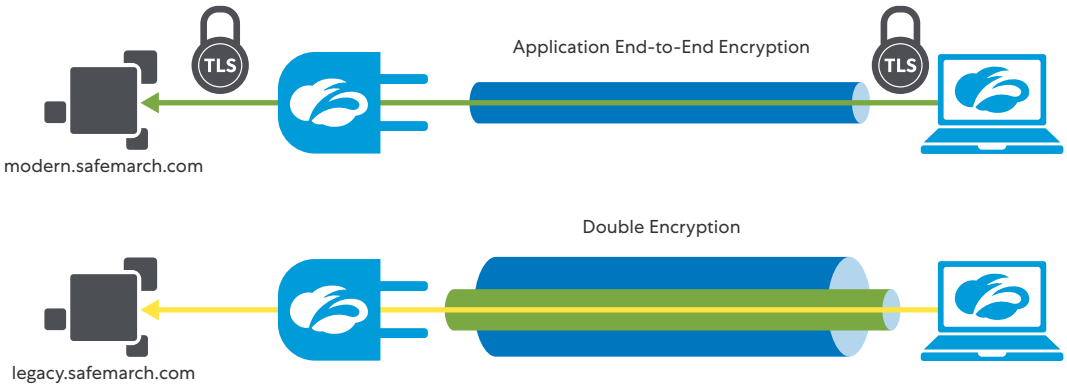


Figure 30: Application end-to-end encryption vs. Double Encryption by Zscaler Client Connector

Double encryption requires some additional certificate work. The Zscaler App Connector and the 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 are seeking 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, will 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 will impact the throughput of the connector. If your throughput falls, you will need to launch additional connectors. Zscaler strongly recommends setting your connectors to Auto Scaling to meet demand in a double encryption deployment. For scaling details, please visit **https://help.zscaler.com/zpa/connector-deployment-prerequisites#ConnectorThroughput**

For more information on double encryption, please visit: **https://help.zscaler.com/zpa/about-double-encryption**

Health reporting

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

The App Connector will use 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 will be merged into the list. Each TCP port and each UDP port will be tested on every listed address.

Example: your application *jira.safemarch.com* returns two IP addresses and 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 will potentially perform 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 will have 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 vs. 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 will take approximately five (5) minutes. As with all monitoring, you should strive to keep the polling interval as short as is reasonable for the best performance.

If your configuration results in more than 6,000 health check targets, all further targets will be 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 will 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 will bypass ZPA. This setting is only appropriate for apps hosted in AWS 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 Direct Connect to the AWS environment – rather than leveraging ZPA's dynamic path optimization to deliver traffic to the application.

The second case deals with applications that may 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 will still attempt 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.

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.

Zscaler, Inc.
120 Holger Way
San Jose, CA 95134
+1 408.533.0288
www.zscaler.com