APPGATE®

AppGate SDP for Azure – Reference Architectures

Type: Technical guide

Date: April 2020

Applies to: AppGate SDP v4.3 or newer

Table of Contents

Introduction

One of the largest perceived barriers to migrating workloads and utilizing cloud infrastructure are security and compliance considerations. Azure of course understands this, and has created innovative capabilities in security management and regulatory compliance frameworks. They have also partnered with many security solutions providers to extend the security capabilities of a cloud infrastructure even further.

Cryptzone is one of those partners. Our product – AppGate – is a leading Software-Defined Perimeter solution available in the marketplace. AppGate provides the following benefits for those deploying on Azure:

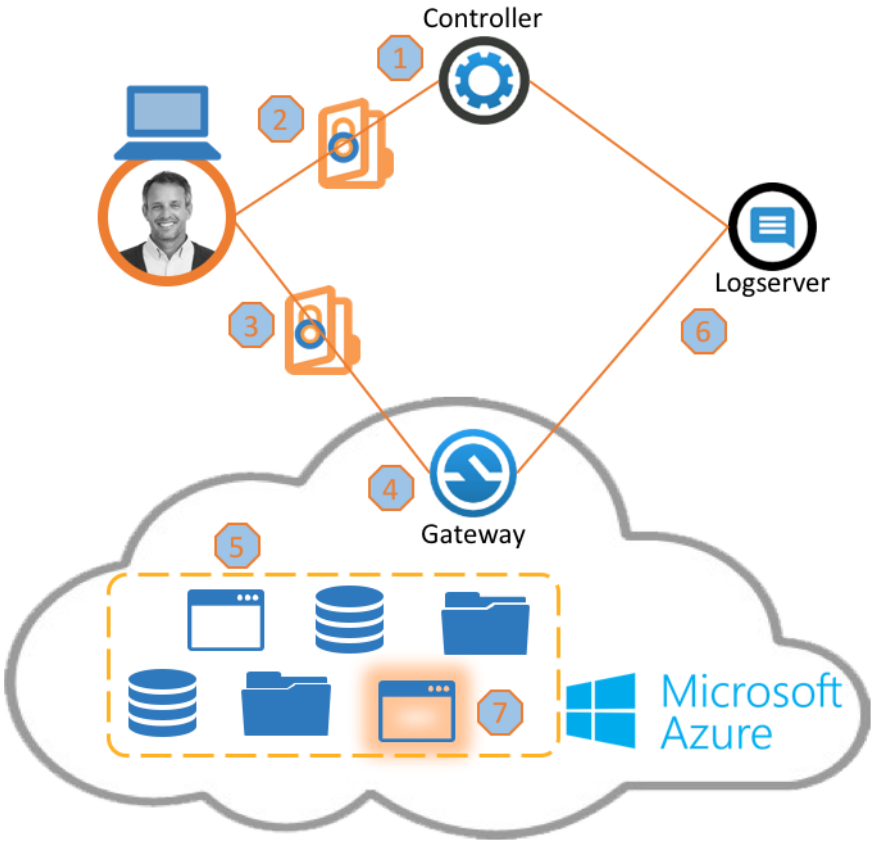
> **Created for the Hybrid Cloud:** AppGate was designed to protect all enterprise cloud resources, including those workloads used in Azure. AppGate has a flexible, distributed deployment model to suit any architecture, automatically detects server instance creation, and leverages user and server attributes to determine access. AppGate also bridges and integrates all elements of the enterprise hybrid cloud infrastructure, controlling access to authenticated users to appropriate cloud resources, wherever they may be.
>
> **Seamless Integrations:** AppGate can reduce cost, complexity and effort for configuring third-party access, privileged user access and cloud infrastructure management. It combines authorization, encryption and access control in one system, replacing many traditional point products. AppGate also integrates with most identity and SIEM solutions, allowing enterprises to take advantage of existing security infrastructure. This enforces strong authentication, and enables organizations to connect network security to their identity management life cycles.
>
> **User-Centric Network Security:** AppGate provides application and service-specific authentication and authorization which controls network access inside and from outside the perimeter. AppGate dynamically creates a secure, encrypted network *segment of one* that's tailored for each user session, based on user attributes. Network access rules aren't written once and saved forever, but are created and enforced in real-time.
>
> **Compliance is Key:** AppGate can help the enterprise reduce regulatory compliance costs by reducing scope and audit complexity. Azure components help with a multitude of regulatory controls, and AppGate can further enhance these. AppGate can also reduce the number systems that fall within audit scope which in turn might eliminate the need for some of the regulatory controls themselves. Robust logging provides all of the possible evidence necessary to meet auditor requirements.

This document provides an introduction to how AppGate might be deployed in Azure, followed by four detailed architecture scenarios.

If you're unfamiliar with the Software-Defined Perimeter (SDP) architecture that AppGate implements, see https://ww2.appgate.com/definitive_guide_to_sdp

How Does AppGate Work (with Azure)?

AppGate utilizes user context to dynamically create a secure, encrypted network 'segment of one' tailored for each user session. The AppGate usage model is the same wherever it is deployed and can be described in a few simple steps:



Usage model

1. User Authentication: The user authenticates to the AppGate Controller, which is optionally connected to an enterprise's IAM system (AD, LDAP, or SAML)
2. Controller Applies Security Policies: The Controller applies policies assigned to the user based on user attributes, roles, and context, and then issues a signed Entitlement token listing the resources the user has access to.
3. User Accesses Resources: The authenticated user can now access protected resources behind a Gateway.
4. Gateway Evaluates User Entitlements: The Gateway evaluates Entitlements in real-time, ensuring that all conditions are met. For example: network location, time of day, device health, or service metadata, such as security groups. Users may be prompted for additional information, such as a one-time password.
5. Gateway Opens Connection to Resource: If the Gateway determines that required conditions have been successfully met, it will open a connection to the protected resource specified by the user.
6. All Actions are logged: Throughout the authentication and authorization process, all actions pertaining to the user, resources, and decisions made by the Controller and Gateways are logged. Logs can be stored or sent to the enterprise SIEM for additional action.

7. Detects New Services: The Gateway constantly monitors for the creation of new services, and – based on this metadata and the allowed Entitlements – can adjust user access as necessary.

Controllers

The Controllers utilize multi-master database synchronization for high availability, based on an eventual consistency concept. Clients typically use the Azure Load balancer to connect to one of the clustered Controllers, but they could use DNS round-robin or the Azure Traffic Manager as well which can distribute the traffic based on Performance, Weighting, Priority or Geography.

The communication between Controllers happens on bi-directional TCP port 444 (mutual TLS connectivity) for control channel messages and bi-directional TCP port 5432 for encrypted mutual database synchronization. The reason why we can use eventual consistency (meaning we don't have to wait until all database have processed the record update), is that there is no session state kept within the database, the state is handled by the signed tokens the Controllers send to the Clients. This means that the Controllers only need to store the configuration, policies and entitlements, and that most database operations will be read operations to generate the secure tokens.

The databases are drawn here outside the Controller, to explain the different encrypted TCP streams, but the database is always an integral part of an Azure Controller instance (virtual appliance).

The Controller is normally connected to an Identity Management system, which serves to validate user authentication and act as the source of user attributes and group memberships. The Identity Management system may be located anywhere, as long as the Controller can access it. AppGate supports AD, LDAP, RADIUS and SAML-based identity systems.

Note that in the case of SAML, the Controller acts as the Relying Party, and the Client device authenticates directly with the SAML IDP, relaying the SAML assertions to the Controller. This follows a slightly modified SAML interaction flow. More information about this is available in a SAML guide, available on request.

Gateways

Clients obtain the list of Gateways in each pool (called "site") from the Controller (after successful authentication), and connect to one of the Gateways in the pool. The Client receives a list of descriptive network entitlements for each site. Each network entitlement token is signed by one of the Controllers so the Gateway will verify the signature before creating a micro private firewall for this specific Client/device combination. The Gateway talks to Azure, to translate descriptive entitlements like Azure://tag:SSH=Linux-administrators into IP addresses. The micro private firewall will now be configured to allow SSH access to all instances that have this tag within the Resource Group. If instances are removed or added with the tag SSH=Linux-administrators, the rules inside the micro private firewall will automatically adapt. AppGate can auto-resolve the instance IP's based on a number of different parameters within Azure. This includes the use of names, tag key or tag value referenced against hosts, Networks Security Groups, Virtual Networks, SubNets and Load Balancers.

Gateways will receive mutual TLS traffic on port 443 from the Clients and will send and receive mutual TLS traffic on port 444 from all Controllers. There is no communication between Gateways whatsoever.

Clients

After installation, the first time the Client device connects to the Controller the device needs to go through an on-boarding procedure.

Single Packet Authorization (SPA) – requires the Client to open the TCP connection with the AppGate system by using a specific pre-shared key.

The Client is required to accept the Controllers certificate as it will be used in this and subsequent communications to verify the authenticity of the Controller. The Client connects on port 443 (TLS) and passes only system control data.

The Client will ask for authentication credentials and optionally a One Time Password (using the built-in or an external Radius service configured by the customer) to on-board the Client device. The device will receive an on-boarding secure token, which glues the user credentials with the device ID. With on-boarding disabled, a user with valid credentials (username and password) will only be able to use their own on-boarded devices.

The Client generates its own private/public key pair and based on this receives a Client Certificate signed by the CA that will be used for all subsequent mutual TLS connections between the Client and Gateways. This traffic from Clients to Gateway comprises both system control data and application data.

Scenarios

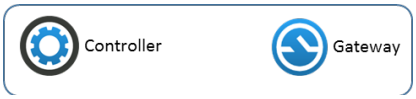
AppGate can be deployed into Azure in a number of different ways. Below are some suggested deployment scenarios, with explanation and analysis.

If you are using Network Security Groups in these scenarios then the default settings might only allow 22 and 443 inbound to any given AppGate appliance. So, you may need to open 444 to allow access between appliances and for admin access.

Best security practice would be to only allow 22 and 444 access to the appliances through the Client tunnel on 443. Effectively using a firewall rule in AppGate that connects to localhost. So, once you are set up you might want to change the Network Security Group rules again to only allow 443 inbound.

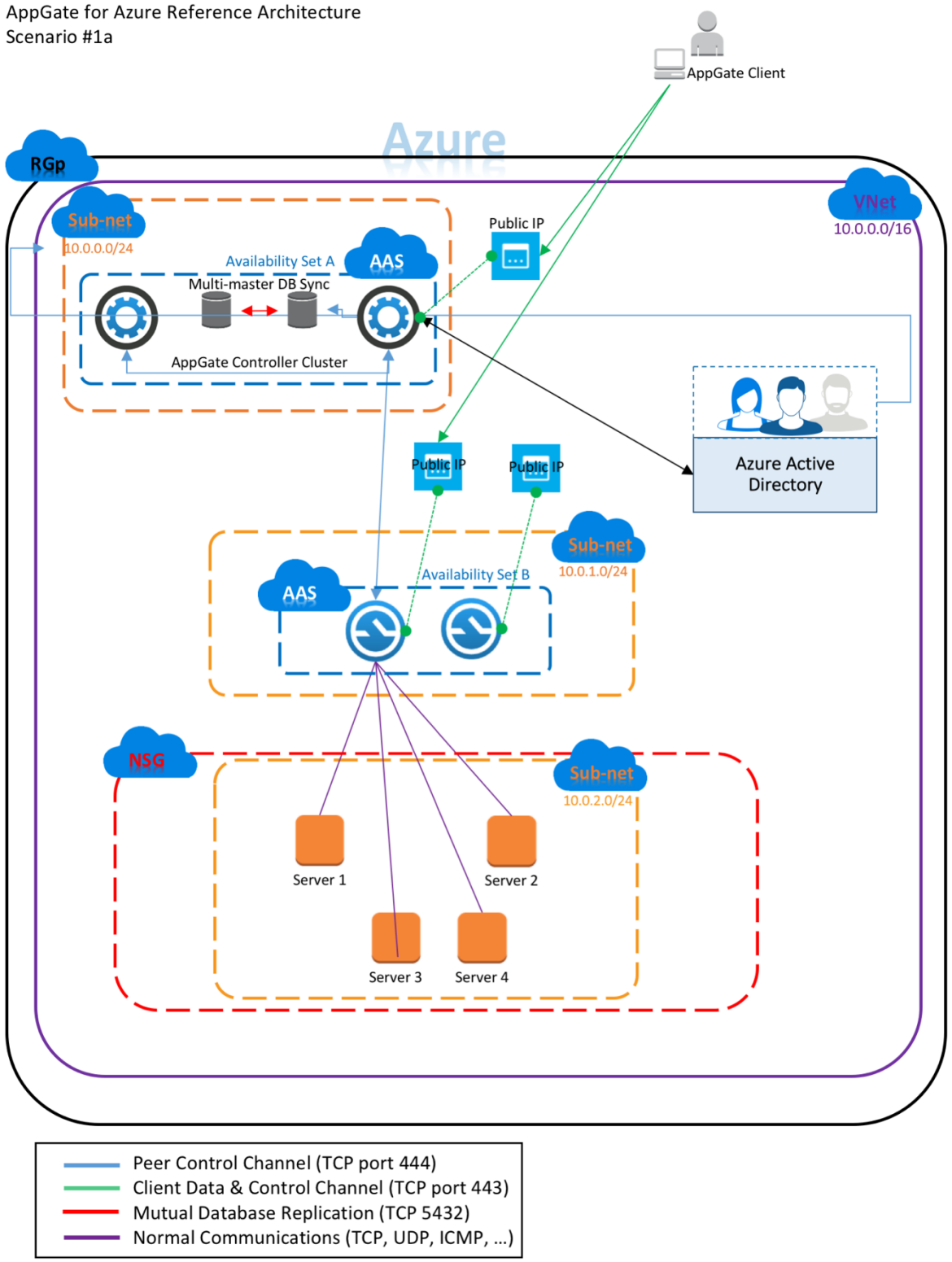
Even though one or two ‘catch-all’ Resource Groups are shown in these scenarios, these don’t really affect the underlying architectures so for the most part these can be ignored.

Note that the diagrams use the following icons to represent AppGate components:



Scenario 1a: AppGate in Azure protecting Location

In this scenario, the entire AppGate system is deployed within Azure. The protected resources sit together in one VNet. The Azure Availability set [AAS] settings can be used to ensure that at least one of the Controllers and one of the Gateways are available all the time.



Scenario 1a - details

Controller:

The AppGate Controller is deployed as a multi-Controller group within a single AAS. Azure ensures that up to 7 VMs in a single AAS are physically separated so that each Controller deployed will suffer failures and receive updates independently from one another. The AAS is a logical concept that overlays the actual operational environment which is defined by other means. Whilst the 2 Controllers are shown as being in the same resource group as the protected hosts, you could use a different one or indeed a different location altogether. It is easiest to have the 2 Controllers in the same VNet otherwise 'site-to-site' links (or the internet) must be used to allow replication. It is also good practice to have a Controller close to the user community to reduce sign-in latency.

Gateway:

In this example, there are 2 Gateways to provide access to the protected servers in the 10.0.2.0/24 sub-net. As with the Controllers, the 2 Gateways sit within a single AAS. As long as one of the Gateways in the AAS is available then you have proper access to the protected servers at all times. Azure best practice suggests that you Group VMs that serve the same purpose into unique AASs; so we have used a different AAS (from the Controllers).

The Gateways are configured with only one interface, sit in their own sub-net and are in front of the protected (Azure) sub-net which contains the protected servers. The Gateways needs to be reachable from outside the Azure network so each have an Azure Public IP assigned to their interface IP address.

All traffic between the AppGate Client and Gateways is encrypted. The Gateways should be set to use NAT, so that the user's traffic appears to be coming from the AppGate's internal IP address when it is forwarded to the protected servers.

Azure is relatively open by default so this scenario works fine without configuring the NSG, however most organizations will probably wish to configure a Network Security Group (NSG) for the hosts in the protected sub-net. Having the Gateways located in a different sub-net (this is not a specific requirement) is not an issue as all traffic inside a VNet is allowed by default; but it will make it easier to configure the NSG. The NSG needs to allow inbound access from the Gateway's IP address and might block all other inbound or outbound traffic from the protected sub-net.

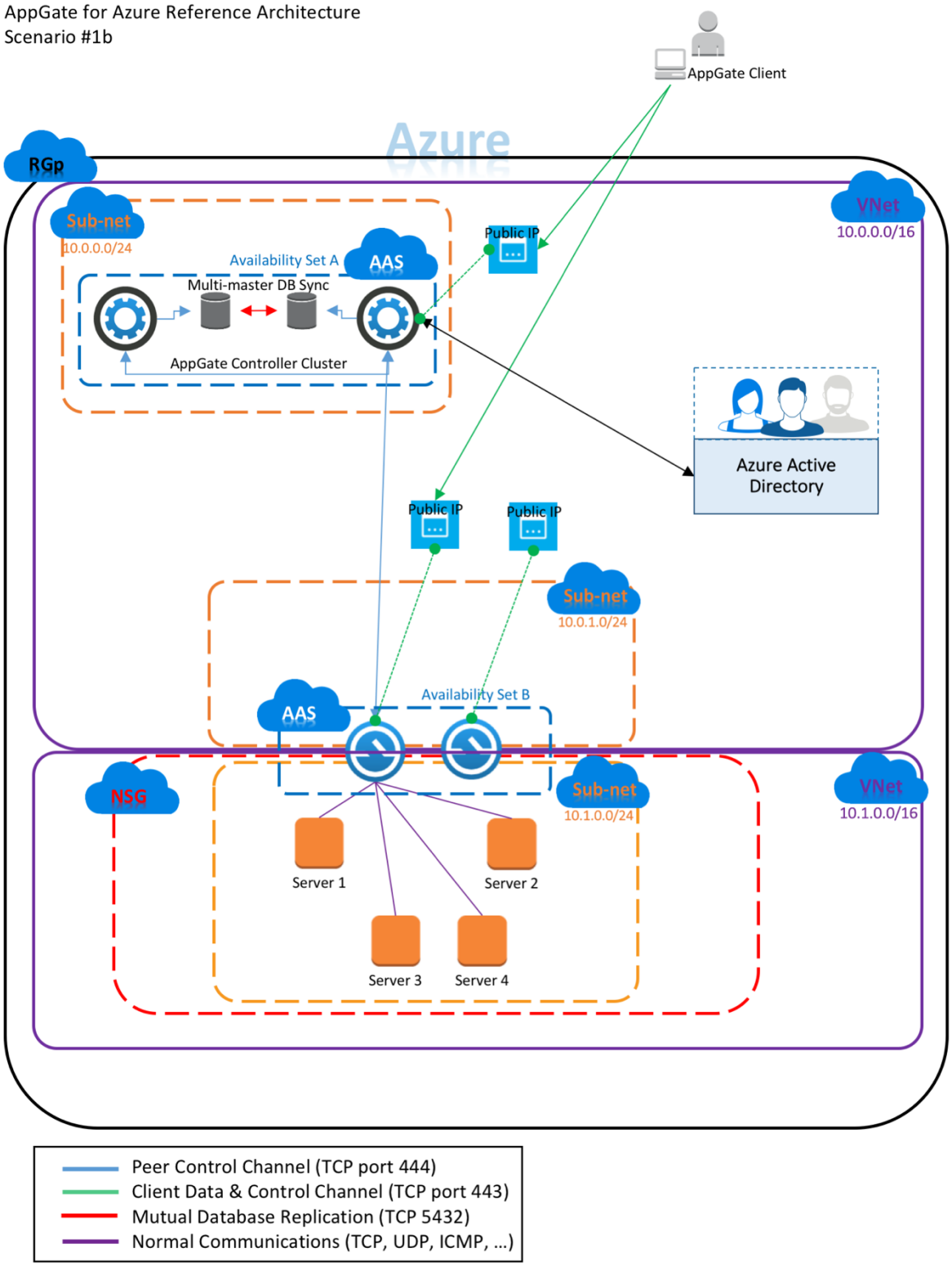
When using Azure it is probably easiest to map the AppGate Site (that protects a defined set of servers), to Azure sub-net(s). You can configure more than one sub-net within a VNet and more Sites within AppGate. This can extend to different VNets and even different locations (as shown in one of the next examples).

Log Server:

Note that the LogServer is omitted from this diagram.

Scenario 1b: AppGate in Azure protecting Location

In this scenario, the entire AppGate system is deployed within Azure. The protected resources sit in a different VNet. The Gateways have 2 interfaces configured – one, internet facing and the other, talking to the protected servers. AAS settings can be used to ensure that at least one of the Controllers and one of the Gateways are available at the same time.



Scenario 1b - details

Controller:

As 1a.

Gateway:

In this example, there are 2 Gateways to provide access to the protected servers in the 10.1.0.0/24 sub-net. As with the Controllers, the 2 gateways sit within a single AAS. As long as one of the Gateways in the AAS is available then you have proper access to the protected servers at all times. Azure best practice suggests that you Group VMs that serve the same purpose into unique AASs; so we have used a different AAS (from the Controllers).

In Azure it is possible to define multiple interfaces on Gateway instances and assign these to different sub-nets. This must be done when the instance is created and not afterwards. In this scenario, the Gateways are configured with two interfaces, so sit between two sub-nets. These sub-nets can be in different VNets or even Resource Groups. In this case the sub-nets are in different VNets which in Azure are logically isolated from one another. This means nothing in the 10.0.1.0/24 sub-net can talk to the 10.1.0.0/24 network unless it passed through the Gateway server – which is a perfect arrangement. The Gateways needs to be reachable from outside the Azure network so each have an Azure Public IP assigned to their internet facing interface.

All traffic between the AppGate Client and Gateways is encrypted. The Gateways should be set to use NAT, so that the user's traffic appears to be coming from the AppGate's 10.1.0.x internal IP address when it is forwarded to the protected servers.

Azure is relatively open by default so this scenario works fine without configuring the NSG, however most organizations will probably wish to configure a Network Security Group (NSG) for the hosts in the protected sub-net. Having one interface of the Gateways located inside the protected sub-net means the NSG rules for the sub-net only need to cater for the protected server as the access rules are now internal to the AppGate system.

When using Azure it is probably easiest to map the AppGate Site (that protects a defined set of servers), to Azure sub-net(s). You can configure more than one sub-net within a VNet and more Sites within AppGate. This can extend to different VNets and even different Locations (as shown in one of the next examples).

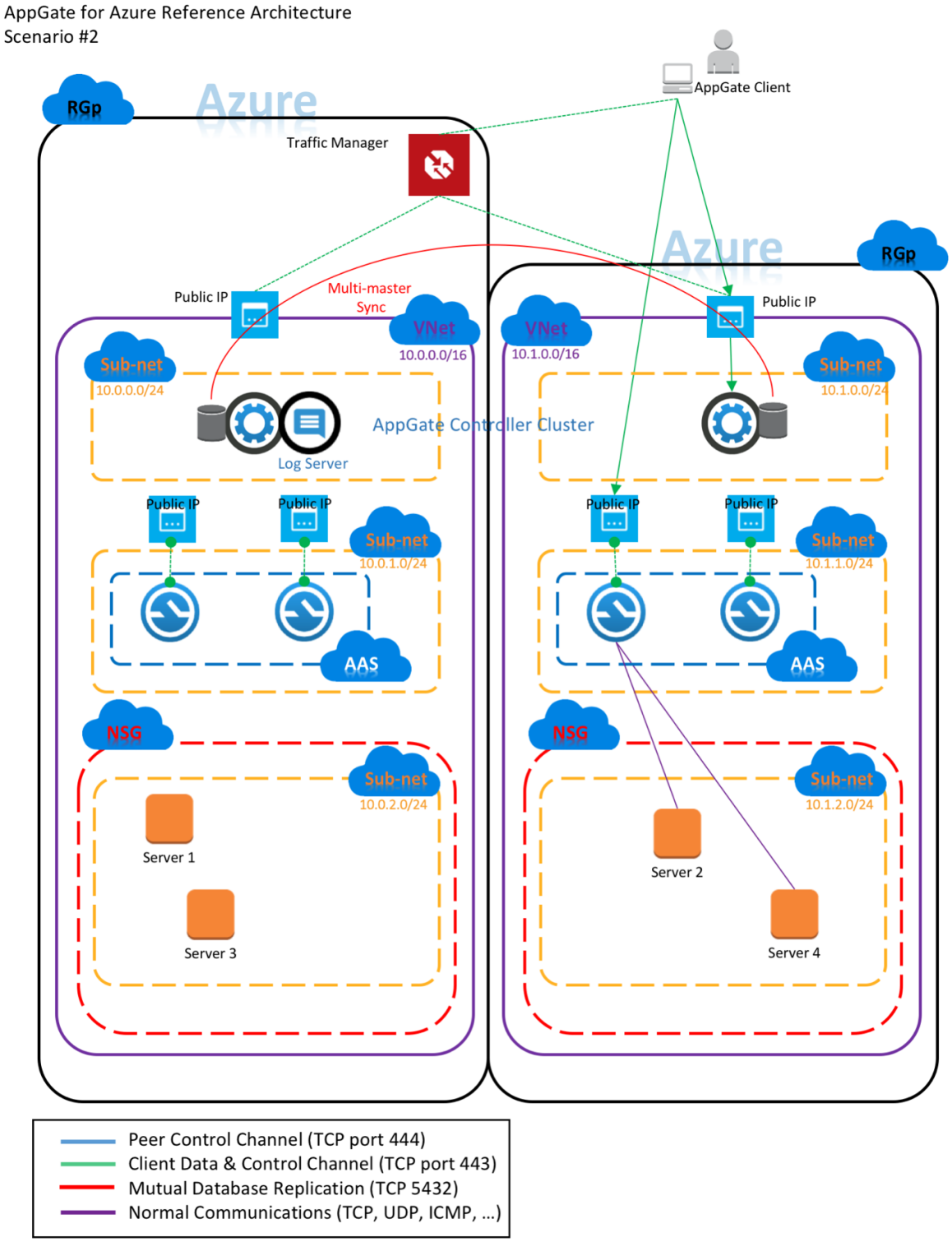
Log Server:

Note that the LogServer is omitted from this diagram.

Scenario 2: AppGate in Azure using different Resource Groups / Locations

Like in scenario 1, the entire AppGate system is deployed within Azure. However, in this scenario there are two Sites, each protected by Gateway pools.

When you create a Resource Group in Azure you define the Location at that time. So, the two Azure Resource Groups might be in the same or different Locations – it makes no difference as far as AppGate is concerned. The Controller cluster also spans multiple Azure Resource Groups each Resource Group can belong to the same or different Azure accounts. AAS can ONLY be defined within one Resource Group so there is no way to place the 2 Controllers into one availability set; so the best way to guarantee availability is to be running in different Locations.



Scenario 2 - details

Controller:

In this scenario, we want to have users connect to the closest of the two location by default. The Controllers are deployed across multiple Azure Resource Groups and Locations. But in this case the normal Azure Load Balancer will not work, as both Controllers reside in a different locations. In this scenario it is possible to take advantage of another Azure resource – the Traffic Manager profile. Traffic Manager applies the selected traffic-routing method to each DNS query it receives. The traffic-routing method determines which Controller will be returned in the DNS response.

> **Priority:** Select 'Priority' when you want to use a primary service endpoint for all traffic, and provide backups in case the primary or the backup endpoints are unavailable.
> **Weighted:** Select 'Weighted' when you want to distribute traffic across a set of endpoints, either evenly or according to weights, which you define.
> **Performance:** Select 'Performance' when you have endpoints in different geographic locations and you want end users to use the "closest" endpoint in terms of the lowest network latency.
> **Geographic:** Select 'Geographic' so that users are directed to specific endpoints (Azure, External or Nested) based on which geographic location their DNS query originates from. This empowers Traffic Manager customers to enable scenarios where knowing a user's geographic region and routing them based on that is important. Examples include complying with data sovereignty mandates, localization of content & user experience and measuring traffic from different regions.

Any of these methods will work for the Controller traffic but it is worth remembering that it is only sign-in information and tokens that are exchanged between the Client and Controller. In this case the Controllers would be selected based on (local) geography so the Client is connecting to its local Controller, the one in the right hand side resource group.

Gateway:

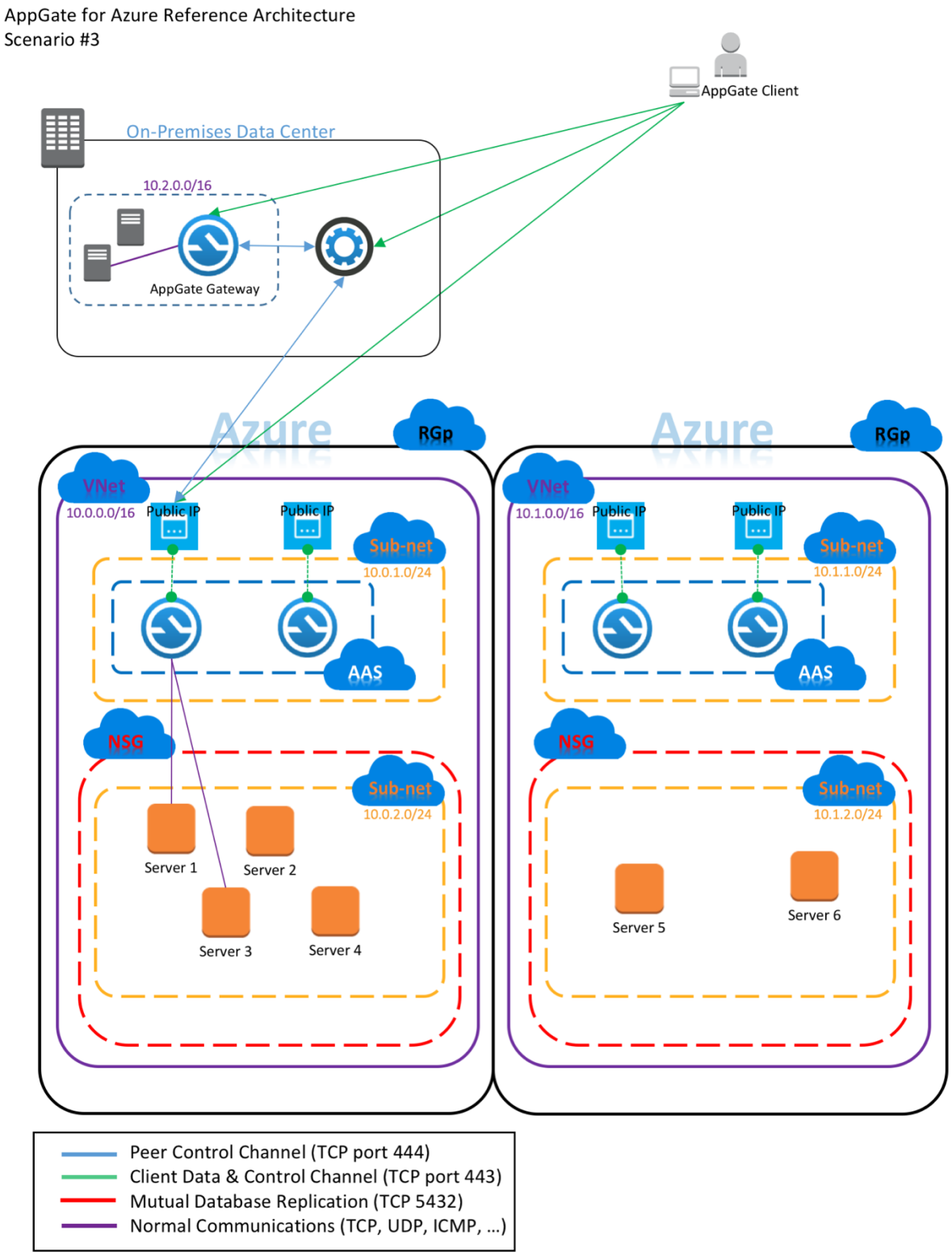
The Gateways are deployed differently – each Site is configured with redundant Gateways to protect the Vnet and its respective sub-net. These would be placed in the same AAS to ensure one was always available. The Gateways do not use the Traffic Manager, instead the Site (Gateways) are chosen by adding a Filter rule that uses the Client's geo claims to select the local geography. In the diagram above, the Client therefore connects to a Gateway running in the right hand side resource group, and its tunneled traffic is directed by the Gateway to servers running in that Vnet.

LogServer:

In this scenario, a single LogServer is shown attached to one Controller. This LogServer collects all log information from all Gateway and Controller instances throughout the entire AppGate collective (connections not shown). Currently, AppGate only supports one appliance acting as a LogServer. In order to support enterprise scenarios, we recommend that each Controller and Gateway appliance be configured to use remote sys log servers (RSYSLOG), for example to a remote SIEM as shown in the diagram. This SIEM would receive logs directly from each Controller and Gateway (SIEM connections from Gateways are omitted for clarity in the diagram)

Scenario 3: AppGate Hybrid Deployment

In this scenario, AppGate is deployed in a hybrid model, with the Controller and a Gateway running on-premises, and with two Gateway-pool protected sites running in Azure.



Scenario 3 - details

In this deployment model, Client interaction with the Controller is identical to that described in scenario 1. This is a good illustration of AppGate's distributed architecture – showing that Controller location is independent from the Gateways and the protected resources behind the Gateways.

Controller:

With the Controller hosted on premises, it is still possible to assign multiple sites both on-premises and inside Azure, all managed by the on-premises Controller.

Gateway:

The Gateways are deployed the same way as in scenario 2 – each Azure Site is configured with redundant Gateways to protect the Vnet and its respective sub-net. But in this case there is no need to add a Filter rule that uses the Client's geo claims to select the local geography.

In this example the Client received Entitlements for an on-premises server, protected by a local Gateway there, and a few Entitlements to access services in the Vnet with subnet 10.0.0.0/16. Since the user/device has no Entitlements for the second Azure Vnet with subnet 10.1.0.0/16, the Client will not establish a mutual connection to that Site. Even within the first Vnet subnet (10.0.0.0/16) the Client doesn't have entitlements for instances 2 and 4, and those instances will be invisible to the Client.

Both tunnels are active at the same time and each Gateway will detect if there are any changes within its Vnet that requires the micro private firewall within Gateway to update his firewall rule sets. Similar as in scenario 1, the Gateways can be contacted over the internet using a Public IP. All traffic to both Gateways will now be encrypted using the Client data channel over port 443 to both Gateways.

Log Server:

Note that the LogServer is omitted from this diagram.

Resources

You'll find additional resources on the AppGate website.

The AppGate SDP product documentation is available here:

- Admin Guide: https://sdphelp.appgate.com
- Client User Guide: https://sdphelp.appgate.com/userguide

Access to our support services (including further articles) is via the customer portal.

Thank you, and we hope you find AppGate SDP to be a valuable solution to your security challenges.