


Securely managed web applications

05/09/2019 • 9 minutes to read • Contributors 

In this article

[Relevant use cases](#)

[Architecture](#)

[Considerations](#)

[Deploy the scenario](#)

[Pricing](#)

[Related resources](#)

This scenario provides an overview of deploying secure applications using the [Azure App Service Environment \(ASE\)](#). To restrict application access from the Internet, the Azure Application Gateway service and Web Application Firewall are used. This article will also provide guidance around continuous integration & continuous deployment (CI/CD) for App Service Environments using Azure DevOps.

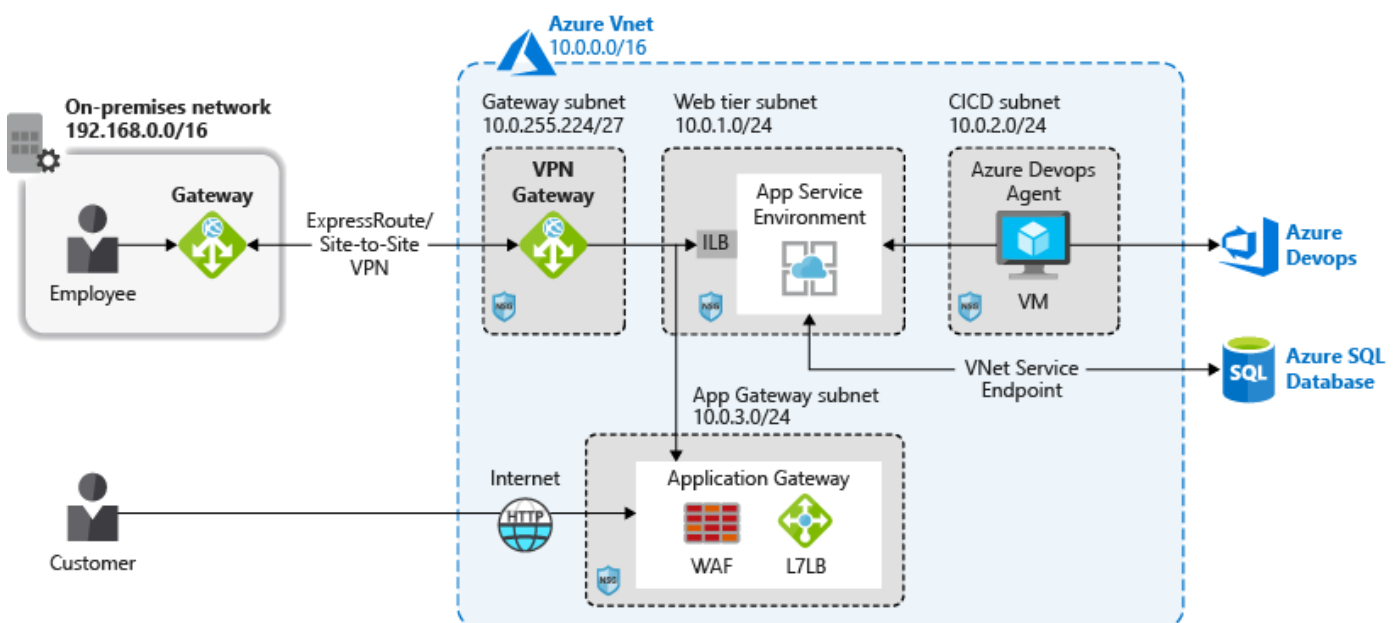
This scenario is commonly deployed in industries such as banking and insurance where customers are conscious of platform level security in addition to application level security. To demonstrate these concepts, we'll use an application that allows users to submit expense reports.

Relevant use cases

Consider this scenario for the following use cases:

- Building an Azure Web App where additional security is required
- Dedicated tenancy is needed, rather than shared tenant App Service Plans
- Utilize Azure DevOps with an [internally load balanced](#) Application Service Environment, often called an ILB ASE.

Architecture



Data flows through the scenario as follows:

1. HTTP/HTTPS requests first hit the Application Gateway

2. Optionally (not shown in the diagram) you can have Azure Active Directory (Azure AD) Authentication enabled for the Web App. After the traffic first hits the Application Gateway, the user is then prompted to supply the credentials to authenticate with the application.
3. User requests flow through the internal load balancer (ILB) of the ASE, which in turn routes the traffic to the Expenses Web App.
4. User then proceeds to create an Expense Report
5. As part of the expense creation, the deployed API App is invoked to retrieve user's manager name and email
6. The Created Expense Report is stored in Azure SQL Database
7. To facilitate continuous deployments, code gets checked into the Azure DevOps instance
8. The build VM has the Azure DevOps Agent installed which allows the build VM to pull the bits for the Web App to deploy to the ASE (as the Build VM is deployed in a subnet inside the same virtual network)

Components

- The [App Service Environment](#) provides a fully isolated and dedicated environment for securely running the application at high scale. In addition, since ASE and the workloads that run on it are behind a virtual network, it also provides an additional layer of security and isolation. The requirement of high scale and isolation drove toward selecting ILB ASE.
- This workload is using the [isolated App Service pricing tier](#) so the application is running in a private, dedicated environment in an Azure datacenter using Dv2-series VMs with faster processors, SSD storage, and double the memory-to-core ratio compared to Standard
- Azure App Services [Web App](#) and [API App](#) host web applications and RESTful APIs. These are hosted on the Isolated Pricing Tier plan that also offers autoscaling, custom domains, and so on, but in a dedicated tier.
- Azure [Application Gateway](#) is a web traffic load balancer operating at Layer 7 that manages traffic to the web application. It offers SSL off-loading, that removes additional overhead on the web servers hosting the web app to decrypt traffic again.
- [Web Application Firewall](#) (WAF) is a feature of Application Gateway. The reason for enabling the WAF in the Application Gateway is to further enhance security. The WAF uses OWASP rules to further protect the web application against attacks such as cross-site scripting, session hijacks, and SQL injection.
- [Azure SQL Database](#) was selected as majority of the data in this application is relational data, with some data as documents and BLOB.
- [Azure Networking](#) provides a variety of networking capabilities in Azure and the networks can further be peered with other virtual networkS in Azure or connectivity can be established with on-premises data centers via Express Route or Site to Site. In this case, a [service endpoint](#) is enabled on the virtual network to ensure the data is flowing just between the Azure virtual network and the SQL Database instance.
- [Azure DevOps](#) is used for teams to collaborate during many sprints, utilizing features of Azure DevOps that support Agile Development, and create build and release pipelines.
- An Azure build [VM](#) was created so that the installed agent can pull down the respective build, and deploy the web app to the ASE environment.

Alternatives

ASE can run regular web apps on Windows or like in this instance, the web apps deployed inside the ASE are each running as Linux containers. ASE was thus selected to host these single instance containerized applications. There are some other alternatives and below is when to consider those platforms when designing your solution.

- [Azure Service Fabric](#) - If you are a Windows-based shop, and the type of workloads are primarily .NET Framework, and you are not yet considering rearchitecting to .NET Core, then use Service Fabric to support and deploy Windows Server Containers. Additionally, Service Fabric supports C# or Java Programming APIs, for developing native microservices, the clusters can be provisioned as Windows or Linux based.
- [Azure Kubernetes Service](#) (AKS) is an Open Source Project and an orchestration platform more suitable to host complex multi-container applications that commonly use a microservices-based architecture. AKS is a managed Kubernetes Azure Service that abstracts away the complexities of provisioning and configuring a Kubernetes

Cluster. However, significant knowledge of Kubernetes platform is still required to support and maintain it, thus to host a handful of single instance containerized web applications may not be the best option.

Other options for the data tier include:

- [Azure Cosmos DB](#): If the majority of the data is in non-relational format, Cosmos DB is a good alternative. This service provides a platform to run other data models such as Mongo DB, Cassandra, Graph data, or simple table storage.

Considerations

There are certain considerations to be aware of when dealing with certificates on ILB ASE. The real trick here is generating a certificate that is chained up to a trusted root without requiring a Certificate Signing Request generated by the server on which the cert will be eventually placed. With IIS for example, the first step is to generate a CSR from your IIS server and then send it to the SSL certificate issuing authority.

You cannot issue a CSR from the Internal Load Balancer (ILB) of an ASE. The way to handle this is to use [this procedure](#).

The above allows you to use proof of DNS name ownership instead of a CSR. If you own a DNS namespace, you can put in special DNS TXT record, the above service checks that the record is there, and if found, knows that you own the DNS server because you have the right record. Based on that, it issues a certificate that is signed up to a trusted root, which you can then upload to your ILB. You don't need to do anything with the individual certificate stores on the Web Apps because you have a trusted root SSL certificate at the ILB.

Make self-signed or internally issued SSL cert work if we want to make secure calls between services running in ILB ASE. Another [solution to consider](#) on how to make ILB ASE work with internally issued SSL certificate and how to load the internal CA to the trusted root store.

While provisioning the ASE consider the following limitations when choosing a domain name for the ASE. Domain Names cannot be:

- net
- azurewebsites.net
- p.azurewebsites.net
- nameofthease.p.azurewebsites.net

Additionally, the custom domain name used for apps and the domain name used by the ILB ASE cannot overlap. For an ILB ASE with the domain name contoso.com, you can't use custom domain names for your apps like:

- [www.contoso.com](#)
- abcd.def.contoso.com
- abcd.contoso.com

Choose a domain for the ILB ASE that won't have a conflict with those custom domain names. You can use something like contoso-internal.com for the domain of your ASE for the example here, because that won't conflict with custom domain names that end in .contoso.com.

Another point to consider is regarding DNS. In order to allow applications within the ASE to communicate with each other, for instance a web application to talk to an API, you will need to have DNS configured for your virtual network holding the ASE. You can either [bring your own DNS](#) or you can use [Azure DNS private zones](#)

Availability

- Consider leveraging the [typical design patterns for availability](#) when building your cloud application.
- Review the availability considerations in the appropriate [App Service web application reference architecture](#)
- For additional considerations concerning availability, see the [availability checklist](#) in the Azure Architecture Center.

Scalability

- Understand how [scale works](#) in ASE
- Best Practices for [cloud apps auto scale](#)
- When building a cloud application be aware of the [typical design patterns for scalability](#).
- Review the scalability considerations in the appropriate [App Service web application reference architecture](#)
- For other scalability topics, see the [scalability checklist](#) available in the Azure Architecture Center.

Security

- Consider leveraging the [typical design patterns for security](#) where appropriate.
- Review the security considerations in the appropriate [App Service web application reference architecture](#).
- Consider following a [secure development lifecycle](#) process to help developers build more secure software and address security compliance requirements while reducing development cost.
- Review the blueprint architecture for [Azure PCI DSS compliance](#).

Resiliency

- Consider using [Geo Distributed Scale with ASE](#) for greater resiliency and scalability.
- Review the [typical design patterns for resiliency](#) and consider implementing these where appropriate.
- You can find a number of [recommended practices for App Service](#) in the Azure Architecture Center.
- Consider using active [geo-replication](#) for the data tier and [geo-redundant](#) storage for images and queues.
- For a deeper discussion on [resiliency](#), see the relevant article in the Azure Architecture Center.

Deploy the scenario

To deploy this scenario, you can follow this [step-by-step tutorial](#) demonstrating how to manually deploy each component. This tutorial also provides a .NET sample application that runs a simple Contoso Expenses reporting application.

Pricing

Explore the cost of running this scenario, all of the services are pre-configured in the cost calculator. To see how the pricing would change for your particular use case change the appropriate variables to match your expected traffic.

We have provided three sample cost profiles based on amount of traffic you expect to get:

- **Small:** This pricing example represents the components necessary to build the out for a minimum production level instance. Here we are assuming a small number of users, numbering only in a few thousand per month. The app is using a single instance of a standard web app that will be enough to enable autoscaling. Each of the other components is scaled to a basic tier that will allow for a minimum amount of cost but still ensure that there is SLA support and enough capacity to handle a production level workload.
- **Medium:** This pricing example represents the components indicative of a moderate size deployment. Here we estimate approximately 100,000 users using the system over the course of a month. The expected traffic is handled in a single app service instance with a moderate standard tier. Additionally, moderate tiers of cognitive and search services are added to the calculator.
- **Large:** This pricing example represents an application meant for high scale, at the order of millions of users per month moving terabytes of data. At this level of usage high performance, premium tier web apps deployed in multiple regions fronted by traffic manager is required. Data consists of the following components: storage, databases, and CDN, are configured for terabytes of data.

Related resources

- [Integrate your ILB App Service Environment with the Azure Application Gateway](#)
- [Integrate your Web Apps with the Azure Application Gateway](#)