## VNet

A single Azure Virtual Network will be created to host and isolate the resources into subnets.

### Subnets

* ag-subnet for Application Gateway with WAF
  + Recommended to have its own subnet ([Source](https://learn.microsoft.com/en-us/azure/virtual-network/vnet-integration-for-azure-services#services-that-can-be-deployed-into-a-virtual-network))
  + Has built in WAF rules
* apim-subnet for API Management
  + Recommended to have its own subnet
  + Deployed in internal mode ([Source](https://learn.microsoft.com/en-us/azure/api-management/api-management-using-with-internal-vnet))
* func-booking-ext-subnet for the HTTP-triggered Azure Function
  + Recommended to have its own subnet
* func-booking-int-subnet for the Queue triggered Durable Functions
  + Recommended to have its own subnet
* asb-subnet for the Azure Service Bus’s Private Endpoint ([Source](https://learn.microsoft.com/en-us/azure/private-link/availability) / [Source](https://learn.microsoft.com/en-us/azure/service-bus-messaging/private-link-service))
  + Can then use Private DNS Zone to resolve the private IP ([Source](https://learn.microsoft.com/en-us/azure/virtual-network/vnet-integration-for-azure-services#private-link-and-private-endpoints))
  + ASB deployed with private access only
  + Uses Azure backbone rather than public internet
* akv-subnet for the Azure Key Vault’s Private Endpoint ([Source](https://learn.microsoft.com/en-us/azure/key-vault/general/private-link-service?tabs=portal))
  + Using AKV to store function keys / connection strings
  + Can then use Private DNS Zone to resolve the private IP
  + Uses Azure backbone rather than public internet
* ampls-subnet for Azure Monitor Private Link Scope’s Private Endpoint ([Source](https://learn.microsoft.com/en-us/azure/azure-monitor/logs/private-link-security))
  + Used for Application Insights and Log Analytics

### Security

#### Network Security Groups

|  |  |  |
| --- | --- | --- |
| **Subnet** | **Inbound** | **Outbound** |
| ag-subnet | Allow port 443 (HTTPS) from Internet | Allow port 443 to apim-subnet |
| apim-subnet | Allow port 443 from ag-subnet | Allow port 443 to func-booking-ext-subnet |
| func-booking-ext-subnet | Allow port 443 from apim-subnet | Allow port 443 to akv-subnet ([Source](https://learn.microsoft.com/en-us/azure/key-vault/general/access-behind-firewall)), ampls-subnet and 5761 (AMQP) to asb-subnet |
| func-booking-int-subnet |  | Allow port 443 to akv-subnet, ampls-subnet, and 5761 to asb-subnet |
| asb-subnet | Allow port 5761 from func-booking-ext-subnet and func-booking-int-subnet |  |
| akv-subnet | Allow port 443 from func-booking-ext-subnet and func-booking-int-subnet |  |
| ampls-subnet | Allow port 443 from func-booking-ext-subnet and func-booking-int-subnet |  |

You could also use 443 for AMQP-over-WebSockets but there is a small overhead compared to using AMQP which is the standard for Azure SDKs ([Source](https://learn.microsoft.com/en-us/azure/service-bus-messaging/service-bus-faq#what-ports-do-i-need-to-open-on-the-firewall--)).

For the NSGs, you could use either the IP of the resource, IP ranges of the subnet, Application Security Groups assigned to the resources, or Service Tags. Out of these, Application Security Groups are the easiest to manage and limit the scope to the specific apps, rather than generic service tags.

#### Managed Identities

Each Azure Function will have a system-assigned managed identity.

The external HTTP-triggered app needs the following roles:

* Storage Contributor on its own storage
* Key Vault Secrets User on AKV ([Source](https://learn.microsoft.com/en-us/azure/key-vault/general/rbac-guide?tabs=azure-cli#azure-built-in-roles-for-key-vault-data-plane-operations))
* Azure Service Bus Data Sender on ASB ([Source](https://learn.microsoft.com/en-us/azure/service-bus-messaging/service-bus-managed-service-identity))
* Monitoring Metrics Publisher on AI ([Source](https://learn.microsoft.com/en-us/azure/role-based-access-control/built-in-roles/monitor#monitoring-metrics-publisher))

The internal queue triggered app needs:

* Storage Contributor on its own storage
* Key Vault Secrets User on AKV
* Azure Service Bus Data Receiver on ASB
* Monitoring Metrics Publisher on AI

## Application Gateway

A diagram of a computer

AI-generated content may be incorrect.

[Source](https://learn.microsoft.com/en-us/azure/architecture/web-apps/api-management/architectures/protect-apis)

By using Application Gateway with a WAF, the application will be protected by common attacks. We can also enable Azure DDoS protection on the VNet.

We can also hide the internal APIM endpoints with public endpoints that can stay the same if the backend changes e.g., api.rixleisure.com.

### Considerations

We could use APIM in external mode (and custom domain) with a NSG ([Source](https://learn.microsoft.com/en-us/azure/api-management/api-management-using-with-vnet)), but this would not provide a WAF ([Source](https://learn.microsoft.com/en-us/azure/web-application-firewall/afds/protect-api-hosted-apim-by-waf)).

For multi-region WAF, then would use Azure Front Door ([Source](https://learn.microsoft.com/en-us/answers/questions/301218/azure-waf-frontdoor-vs-azure-waf-application-gatew)).

## APIM

By using APIM, we can ensure that bookings are made by an authenticated user with a Customer role.

We could also implement rate limiting, potentially based on an authenticated user’s id or by IP address ([Source](https://learn.microsoft.com/en-us/azure/api-management/rate-limit-by-key-policy#example)).

Restrict the origin URLs to our website to prevent other sites using our booking API from within the browser ([Source](https://learn.microsoft.com/en-us/azure/api-management/cors-policy#example)).

Future options to A/B test APIs

## AKV

It is recommended my Microsoft to turn of soft-delete and purge protection ([Source](https://learn.microsoft.com/en-us/azure/key-vault/general/soft-delete-overview)).

## Azure Functions

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Supports Linux Containers [Source](https://learn.microsoft.com/en-us/azure/azure-functions/functions-scale) | VNet Non-HTTP Event Scaling [Source](https://learn.microsoft.com/en-us/azure/azure-functions/functions-networking-options?tabs=azure-portal#virtual-network-triggers-non-http) | Inbound IP restrictions  [Source](https://learn.microsoft.com/en-us/azure/azure-functions/functions-networking-options) | Inbound Private Endpoints | VNet integration | Outbound IP restrictions |
| Flex Consumption Plan |  | Y | Y | Y | Y | Y |
| Consumption Plan |  |  | Y |  |  |  |
| Premium Plan | Y | Y | Y | Y | Y | Y |
| Dedicated Plan (App Service Environment) | Y |  | Y | Y | Y (and supports gateway-required VNet integration) | Y |
| Container Apps | Y |  |  |  | Y | Y |

From the above, if we’re wanting a cheaper solution and not expecting high demand, the Flex Consumption Plan, although limiting in not being able to use Linux Containers, enables VNet integrations and using private endpoints to talk to ASB and AKV as well as being a pay-as-you-go model.

My plan is to have an always ready HTTP triggered function (which could be scaled based on usage) that simply sends off the booking as a message to ASB. This means from a user’s perspective, they get a response quick from no warming up and early validation. A potential improvement would be to have rudimentary validation on this request to fail early.

I would then have a service bus triggered function consuming the message from the ASB which has message retry built in ([Source](https://learn.microsoft.com/en-us/azure/azure-functions/functions-bindings-storage-queue-trigger?tabs=python-v2%2Cisolated-process%2Cnodejs-v4%2Cextensionv5&pivots=programming-language-csharp#poison-messages)). It will start a Durable Function. The orchestrator can then implement backoff and retry per individual function ([Source](https://learn.microsoft.com/en-us/azure/azure-functions/durable/durable-functions-error-handling?tabs=csharp-isolated#automatic-retry-on-failure)), and each function can scale (e.g., validate, booking, payment call, email send) ([Source](https://learn.microsoft.com/en-us/azure/azure-functions/flex-consumption-plan#per-function-scaling)).

#### Backoff and Retry

The backoff retry of sub-functions should have a backoff coefficient set as to not overwhelm downstream functions. Unfortunately, the built in options don’t appear to have jitter support or circuit breaker and Polly cannot be used in an Orchestrator. The built in method can prevent a retry if it shouldn’t call again ([Source](https://www.tpeczek.com/2021/09/handling-transient-errors-in-durable.html#:~:text=There%20is%20a%20property%20on%20RetryOptions%20which%20allows%20for%20providing%20a%20callback%20to%20determine%20whether%20an%20activity%20should%20be%20retried.)).

To prevent functions running for longer than expected, a reasonable timeout could also be set ([Source](https://learn.microsoft.com/en-us/azure/azure-functions/durable/durable-functions-error-handling?tabs=csharp-isolated#function-timeouts)) in the future.

By using Durable Functions, it does bring the risk of duplicate booking where:

1. Booking was successful
2. Payment Call fails every backoff retry in the orchestrator
3. The orchestrator then fails
4. The message is retried
5. Booking is then called a second time

To solve this, booking would need to check that a previous successful booking with the same reference has not been made OR we don’t have an orchestration function, and each function sends out a message which is picked up by the next.

Personally, I would go with option 1 as otherwise the workflow is hard to follow and maintain as opposed to having a safety check in booking.

There is also a risk that after multiple message retries, the message dead-letters. This should have alerts setup in Azure Monitor to manually handle.

## Monitoring

Azure Monitor, Log Analytics, and Azure Application Insights are to be deployed in ampls-subnet, with an Azure Monitor Private Link Scope’s Private Endpoint to access them.

These will be used to store metrics, traces, and logs from the Azure Functions.

#### Azure Monitor

Azure Monitor comes with AG, APIM, ASB, and AKV to audit access attempts and policy enforcements. These could optionally also send their logs and metrics to our ampls-subnet.

Custom alerts should be configured for these e.g., on the Service Bus for when messages are in the dead-letter queue and if they repeatedly fail.

## Other Notes

The app will be on .NET 9 as it is the latest and supports ends at the same time as the current LTS (.NET 8) ([Source](https://dotnet.microsoft.com/en-us/platform/support/policy/dotnet-core#lifecycle)).