**Azure AI containers**

Containerization is an approach to software distribution in which an application or service, including its dependencies & configuration, is packaged together as a container image. With little or no modification, a container image can be deployed on a container host. Containers are isolated from each other and the underlying operating system, with a smaller footprint than a virtual machine. Containers can be instantiated from container images for short-term tasks and removed when no longer needed.

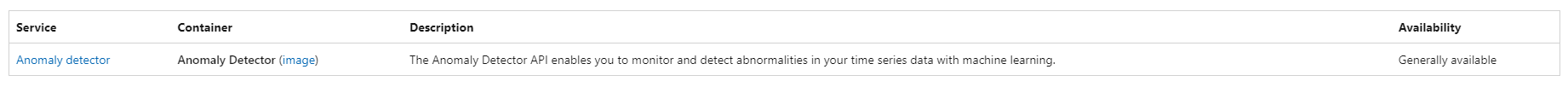
**Features and benefits**

* **Immutable infrastructure**: Enable DevOps teams to leverage a consistent and reliable set of known system parameters, while being able to adapt to change. Containers provide the flexibility to pivot within a predictable ecosystem and avoid configuration drift.
* **Control over data**: Choose where your data gets processed by Azure AI services. This can be essential if you can't send data to the cloud but need access to Azure AI services APIs. Support consistency in hybrid environments – across data, management, identity, and security.
* **Control over model updates**: Flexibility in versioning and updating of models deployed in their solutions.
* **Portable architecture**: Enables the creation of a portable application architecture that can be deployed on Azure, on-premises and the edge. Containers can be deployed directly to [Azure Kubernetes Service](https://learn.microsoft.com/en-us/azure/aks/), [Azure Container Instances](https://learn.microsoft.com/en-us/azure/container-instances/), or to a [Kubernetes](https://kubernetes.io/) cluster deployed to [Azure Stack](https://learn.microsoft.com/en-us/azure-stack/operator). For more information, see [Deploy Kubernetes to Azure Stack](https://learn.microsoft.com/en-us/azure-stack/user/azure-stack-solution-template-kubernetes-deploy).
* **High throughput / low latency**: Provide customers the ability to scale for high throughput and low latency requirements by enabling Azure AI services to run physically close to their application logic and data. Containers don't cap transactions per second (TPS) and can be made to scale both up and out to handle demand if you provide the necessary hardware resources.
* **Scalability**: With the ever growing popularity of containerization and container orchestration software, such as Kubernetes; scalability is at the forefront of technological advancements. Building on a scalable cluster foundation, application development caters to high availability.

**Containers in Azure AI services**

* Azure AI containers provide the following set of Docker containers, each of which contains a subset of functionality from services in Azure AI services. You can find instructions and image locations in the tables below.

**Decision containers**



**Language containers**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

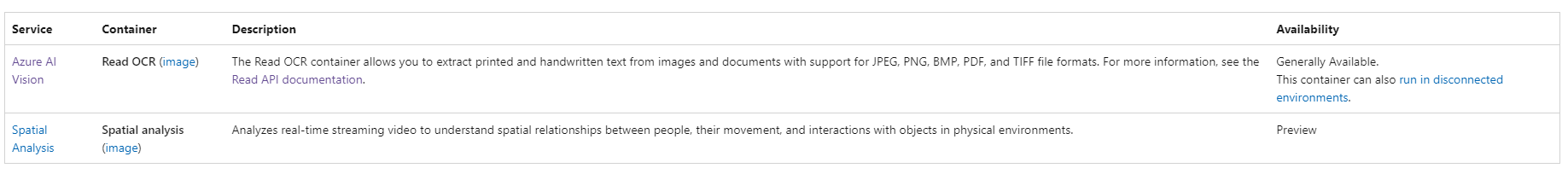
Description automatically generated

**Speech containers**

A screenshot of a computer

Description automatically generated

**Vision containers**



Additionally, some containers are supported in the [Azure AI services multi-service resource](https://learn.microsoft.com/en-us/azure/ai-services/multi-service-resource?pivots=azportal) offering. You can create one single Azure AI services resource and use the same billing key across supported services for the following services:

* Azure AI Vision
* LUIS
* Language service

**Prerequisites**

You must satisfy the following prerequisites before using Azure AI containers:

**Docker Engine**: You must have Docker Engine installed locally. Docker provides packages that configure the Docker environment on [macOS](https://docs.docker.com/docker-for-mac/), [Linux](https://docs.docker.com/engine/installation/#supported-platforms), and [Windows](https://docs.docker.com/docker-for-windows/). On Windows, Docker must be configured to support Linux containers. Docker containers can also be deployed directly to [Azure Kubernetes Service](https://learn.microsoft.com/en-us/azure/aks/) or [Azure Container Instances](https://learn.microsoft.com/en-us/azure/container-instances/).

Docker must be configured to allow the containers to connect with and send billing data to Azure.

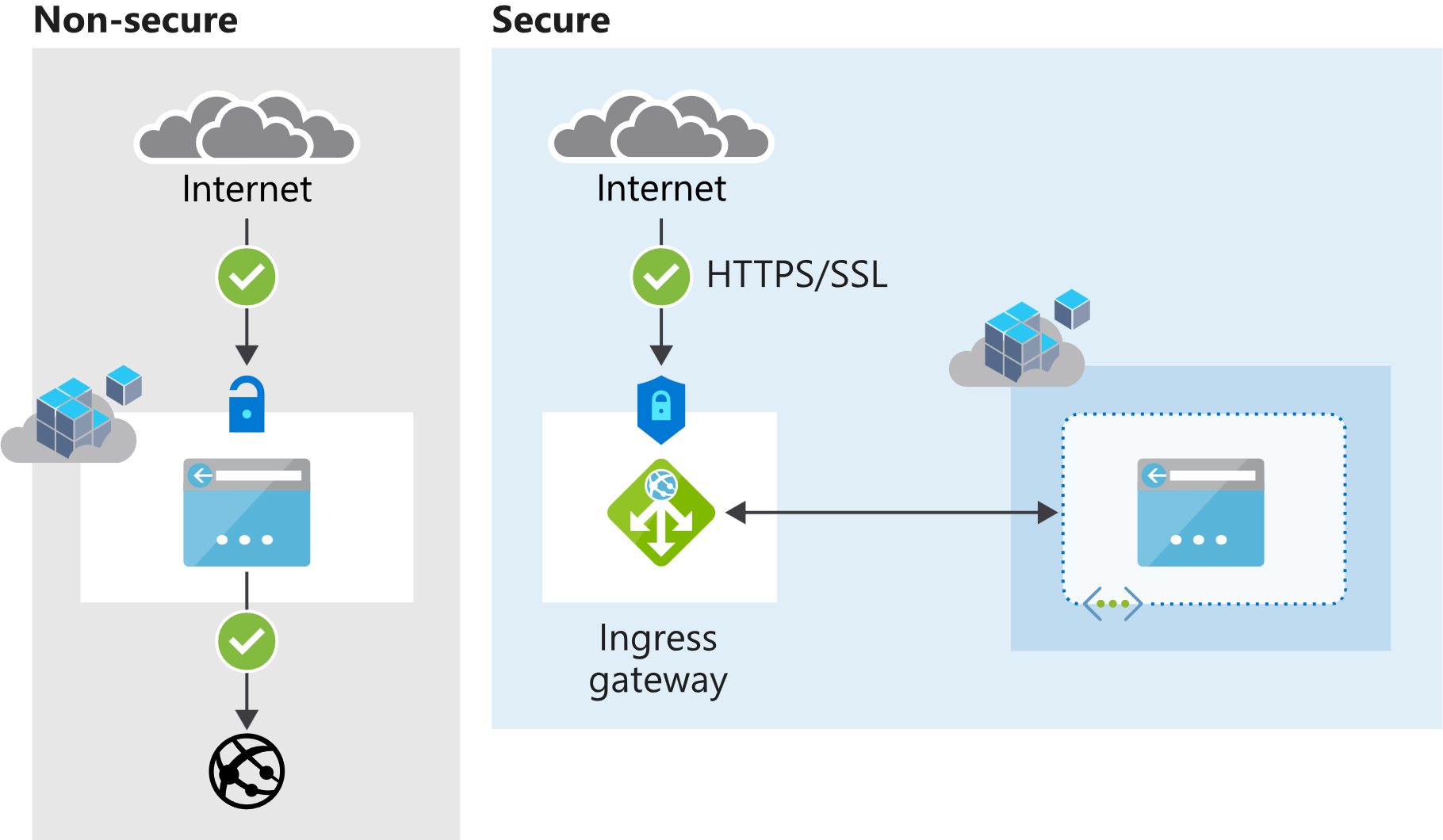
**Familiarity with Microsoft Container Registry and Docker**: You should have a basic understanding of both Microsoft Container Registry and Docker concepts, like registries, repositories, containers, and container images, as well as knowledge of basic docker commands.

Individual containers can have their own requirements, as well, including server and memory allocation requirements.

**Azure AI services container security**

Security should be a primary focus whenever you're developing applications. The importance of security is a metric for success. When you're architecting a software solution that includes Azure AI containers, it's vital to understand the limitations and capabilities available to you. For more information about network security, see [Configure Azure AI services virtual networks](https://learn.microsoft.com/en-us/azure/ai-services/cognitive-services-virtual-networks).

The following diagram illustrates the default and **non-secure** approach:



As an example of an alternative and *secure* approach, consumers of Azure AI containers could augment a container with a front-facing component, keeping the container endpoint private. Let's consider a scenario where we use [Istio](https://istio.io/) as an ingress gateway. Istio supports HTTPS/TLS and client-certificate authentication. In this scenario, the Istio frontend exposes the container access, presenting the client certificate that is approved beforehand with Istio.

[Nginx](https://www.nginx.com/) is another popular choice in the same category. Both Istio and Nginx act as a service mesh and offer additional features including things like load-balancing, routing, and rate-control.

**Container networking**

The Azure AI containers are required to submit metering information for billing purposes. Failure to allowlist various network channels that the Azure AI containers rely on will prevent the container from working.

**Allowlist Azure AI services domains and ports**

The host should allowlist **port 443** and the following domains:

* \*.cognitive.microsoft.com
* \*.cognitiveservices.azure.com

**Disable deep packet inspection**

[Deep packet inspection (DPI)](https://en.wikipedia.org/wiki/Deep_packet_inspection) is a type of data processing that inspects in detail the data sent over a computer network, and usually takes action by blocking, rerouting, or logging it accordingly.

Disable DPI on the secure channels that the Azure AI containers create to Microsoft servers. Failure to do so will prevent the container from functioning correctly.

**Developer samples**

Developer samples are available at our [GitHub repository](https://github.com/Azure-Samples/cognitive-services-containers-samples).

Install and explore the functionality provided by containers in Azure AI services:

* [Anomaly Detector containers](https://learn.microsoft.com/en-us/azure/ai-services/anomaly-detector/anomaly-detector-container-howto)
* [Azure AI Vision containers](https://learn.microsoft.com/en-us/azure/ai-services/computer-vision/computer-vision-how-to-install-containers)
* [Language Understanding (LUIS) containers](https://learn.microsoft.com/en-us/azure/ai-services/luis/luis-container-howto)
* [Speech Service API containers](https://learn.microsoft.com/en-us/azure/ai-services/speech-service/speech-container-howto)
* [Language service containers](https://learn.microsoft.com/en-us/azure/ai-services/language-service/overview#deploy-on-premises-using-docker-containers)
* [Translator containers](https://learn.microsoft.com/en-us/azure/ai-services/translator/containers/translator-how-to-install-container)