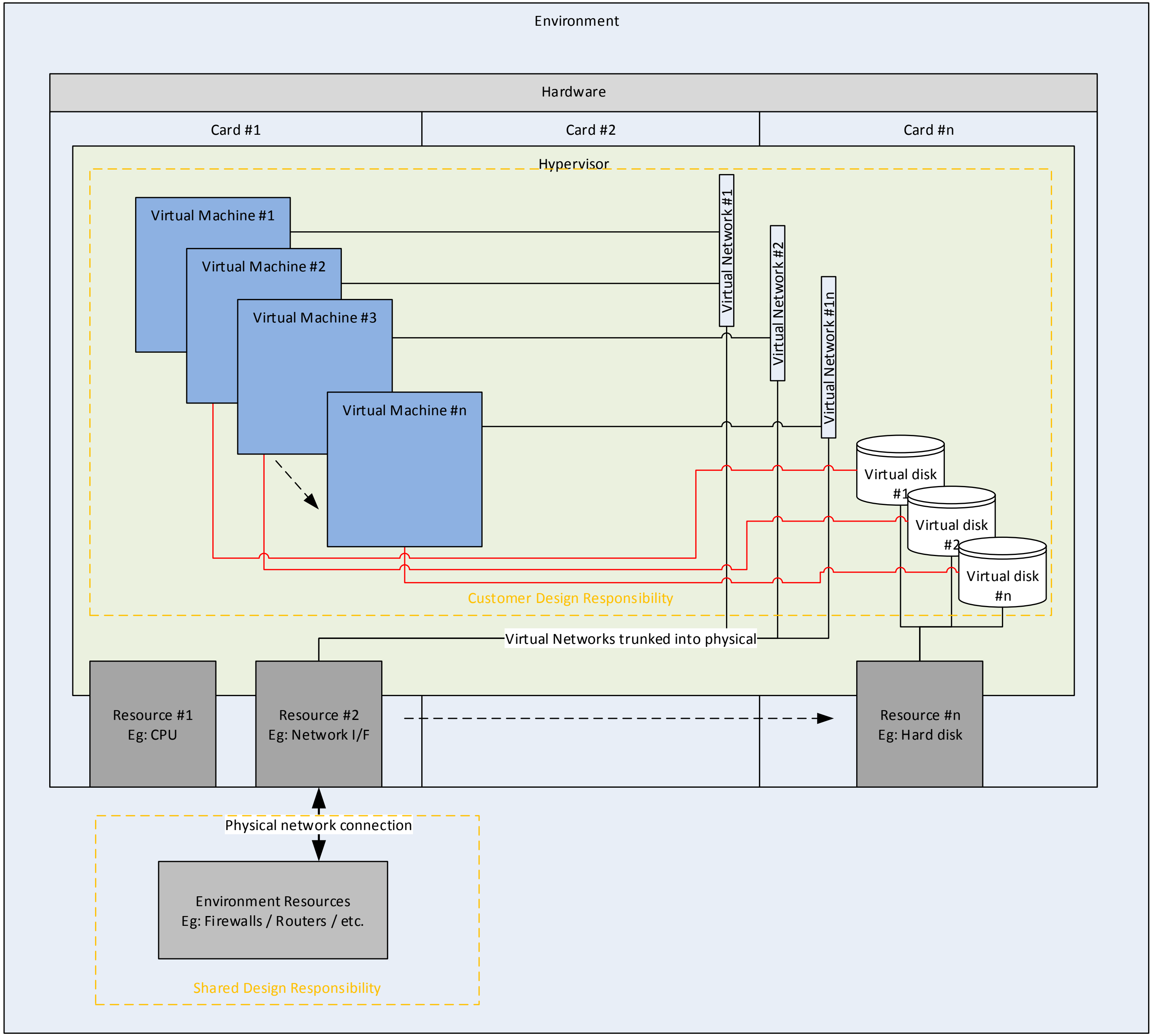


Notes:

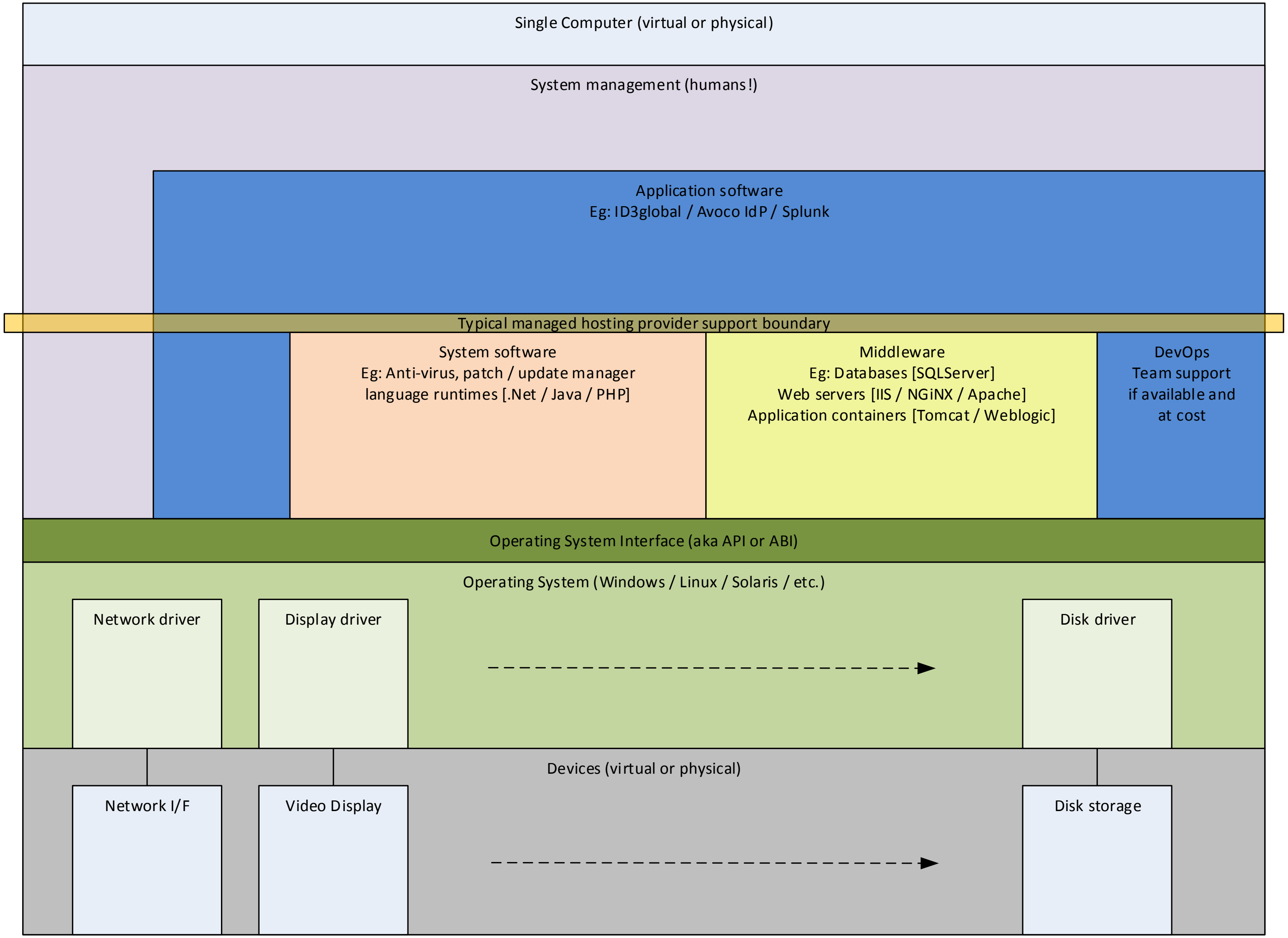
- Hosting company customers are responsible for the design of all the services within the dotted area. Hosting companies will usually offer their advice, but will not be responsible for outcomes.
- Hosting companies will share the design of the environmental components, typically they will manage all hardware and operation while delegating configuration responsibility to their customers.

- Environment covers equipment and services outside the system being designed. This boundary is usually part of a contract, eg:
  - Network connections to Internet or elsewhere.
  - Power and cooling / physical racking.
  - Hosting provider services (eg: rackers access).
- There is a complex data centre architecture that exists outside this view of a system to provide the Environment as agreed in contacts.

- Hardware resources are typically provided by using a number of ‘commodity’ computing cards from the likes of Dell or HP connected together in a card frame and then into racks. These racks will typically contain:
  - processors
  - memory
  - network interfaces
  - storage devices (hard disk, SSD)
  - (sometimes esoteric things like graphics cards)

- Hypervisor software aggregates and virtualises the hardware into a set of resource pools, hiding the physical reality from the application software, and allowing system administrators to allocate fractions of each resource pool into a virtual computer and network infrastructure.
- Hypervisor software can be configured to overcommit (aka lie) about what’s available to make better use of the hardware, it can also dynamically move VMs around if real hardware fails.

- Each virtual machine is a complete computer containing an operating system, applications and data as required to implement the system being designed. More details on the next tab!



Single Computer (virtual or physical)

System management (humans!)

Application software
Eg: ID3global / Avoco IdP / Splunk

Typical managed hosting provider support boundary

System software
Eg: Anti-virus, patch / update manager
language runtimes [.Net / Java / PHP]

Middleware
Eg: Databases [SQLServer]
Web servers [IIS / NGiNX / Apache]
Application containers [Tomcat / Weblogic]

DevOps
Team support
if available and
at cost

Operating System Interface (aka API or ABI)

Operating System (Windows / Linux / Solaris / etc.)

Network driver

Display driver

Disk driver

Devices (virtual or physical)

Network I/F

Video Display

Disk storage

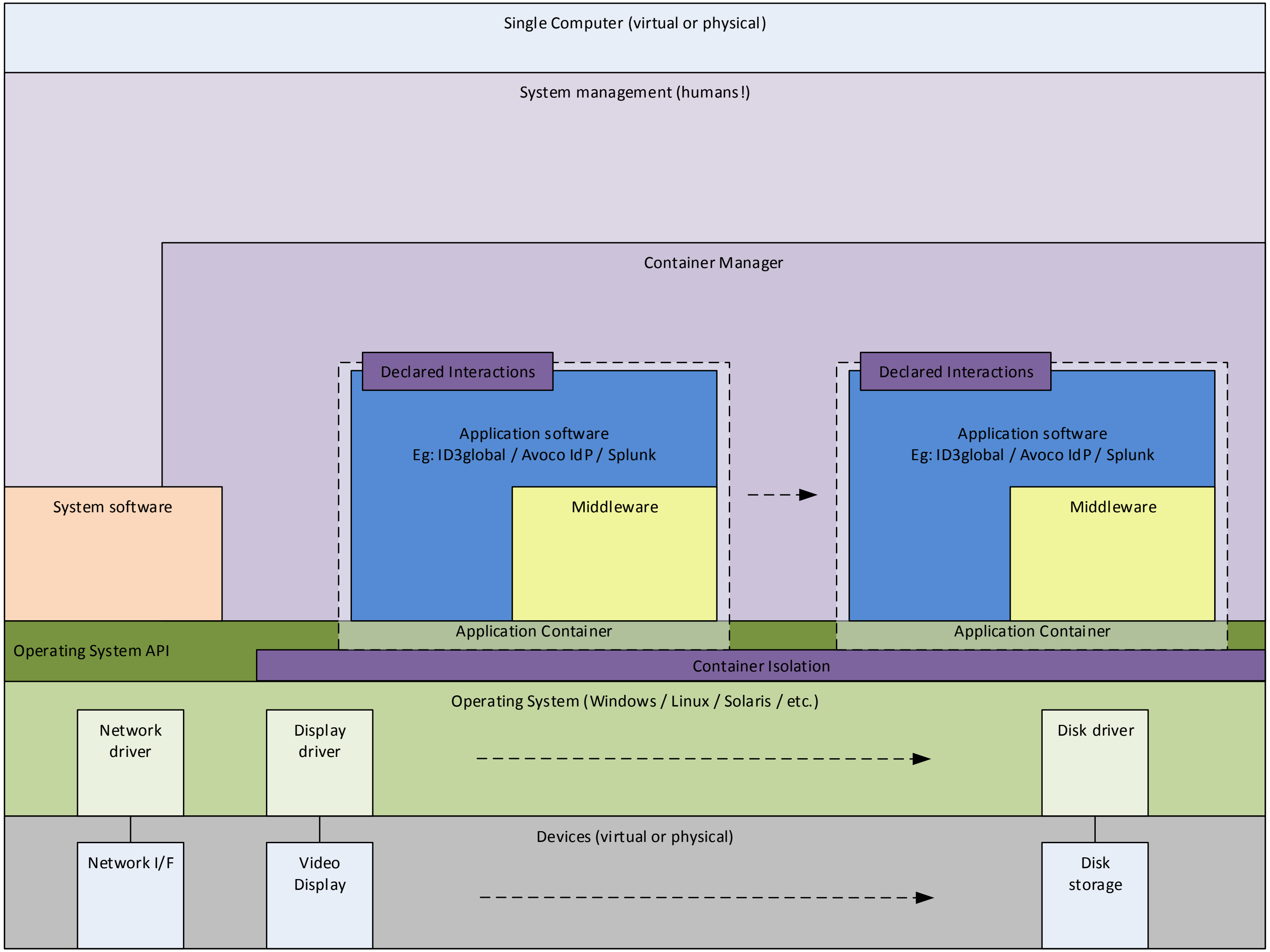
Notes:

- Device driver software in the operating system is typically aware of the virtualization that the Hypervisor provides, as it can operate more efficiently with this knowledge.

- Application software may rely on other system software to function (eg: Avoco IdP relies on both PHP and the NGiNX web server) or it may interact directly with the operating system (eg: installation tools, log monitoring)

- System management by humans involves their interaction with both system software for system management (eg: updates) and application software for application management (eg: Citizen credential resets)

- Hosting providers that offer managed services will usually draw a line under any Application software and only support below that line. Optionally they support popular off-the-shelf middleware such as SQLServer, IIS and NGiNX.
- Where providers offer additional 'DevOps' capabilities they will assign a team to learn the application and support that (at a cost), eg: ProAct support the ID3global application for some activities (daily checks, monitoring) but not complex work such as upgrades.



Notes:

- Containerisation is a form of virtualisation that ***isolates applications and all their dependant middleware and libraries from the execution environment.***

- Containerisation is achieved via a **container manager** component, a related **container isolation** layer below the Operating System API and an **application container** packaging process.

- Containerised applications **declare all interactions** that the contained software has with other resources (eg: processor, memory and storage requirements, network connectivity) via the application container package. The container manager can thus allocate resources to a container and configure the container isolation layer to connect those allocated resources to the container **without having to re-configure the contained software inside**.

- Containerisation provides application portability both within a single computer, or across a cluster of computers (eg: a Data Centre, a Cloud platform), much like virtual machines do, but with finer-grained control of resources and a richer set of resources (eg: network connectivity).

- Several popular cloud hosting platforms support container hosting (eg: Azure, AWS, Rackspace), and frameworks exist to support multi-cloud provider container management (eg: Kubernetes)

- Containers are also deployed locally for development and testing, where the contained software will experience a similar environment to production and both development and testing will be more effective.

- As the container manager automatically configures all the required resources and interconnections using a resource scheduling algorithm, no humans need to be involved in software deployment, increasing operational efficiency, reducing human error.