Secure Programming

A.A. 2022/2023

Corso di Laurea in Ingegneria delle Telecomnicazioni

G. Architectures & Processes

Paolo Ottolino

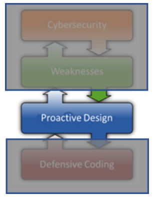
Politecnico di Bari

DIPARTIMENTO DI
INGEGNERIA ELETTRICA
E DELL'INFORMAZIONE

Secure Programming Lab: Course Program

A. **Intro Secure Programming: «Who-What-Why-When-Where-How»**
B. **Building Security in: Buffer Overflow, UAF, Command Inection**
C. **SwA: Weaknesses, Vulnerabilities, Attacks**
D. **SwA (Software Assurance): Vulnerabilities and Weaknesses (CVE, OWASP, CWE)**
E. **Security & Protection: Objectives (CIA), Risks (Likelihood, Impact), Rating Methodologies**
F. **Security & Protection: Security Indicators, BIA, Protection Techniques (AAA, Listing, Duplication etc.)**
G. **Architecture and Processes: App Infrastructure, Cloud, Containers, Orchestration**
H. **Architecture and Processes 2: Ciclo di Vita del SW (SDLC), DevSecOps (OWASP DSOMM, NIST SSDF)**
I. **Free Security Tools: OWASP (ZAP, ESAPI, etc), NIST (SAMATE, SARD, SCSA, etc), SonarCube, Jenkins**
J. **Dynamic Security Test: VA, PT, DAST (cfr. VulnScanTools), WebApp Sec Scan Framework (Arachni, SCNR) :**
K. **Operating Environment: Kali Linux on WSL**
L. **Python: Powerful Language for easy creation of hacking tools**
M. **Exercises: SecureFlag**

Architectures & Processes 1



G.1 App Environment: ZTA Pillars and CSMA Tools.

G.2 Cloud: IaaS, PaaS, CaaS

G.3 Containers: Introduction, Chroot, Domains

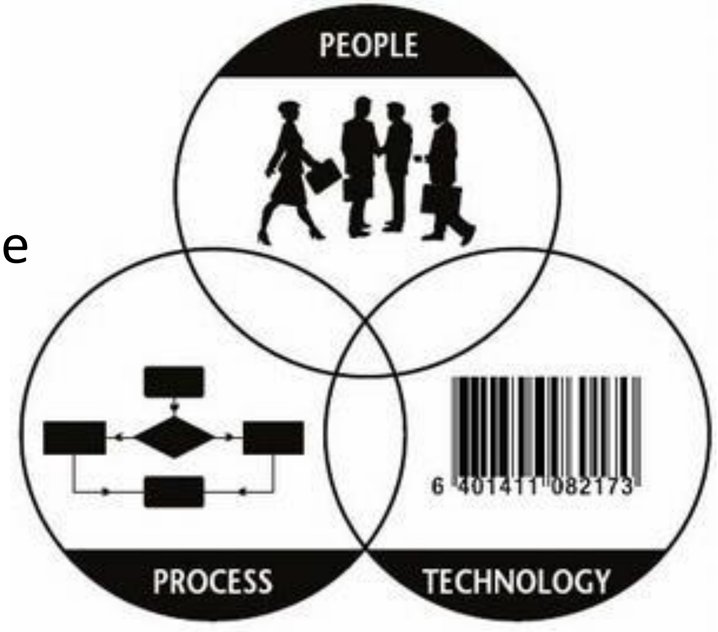
G.4 Orchestration: Manifesto, Phases, Maturity, Tools

G.1 App Environment: Logical Components

People, Process, Technology

People: stakeholders (employees, customers, partners, suppliers)

Process: action performed to execute business

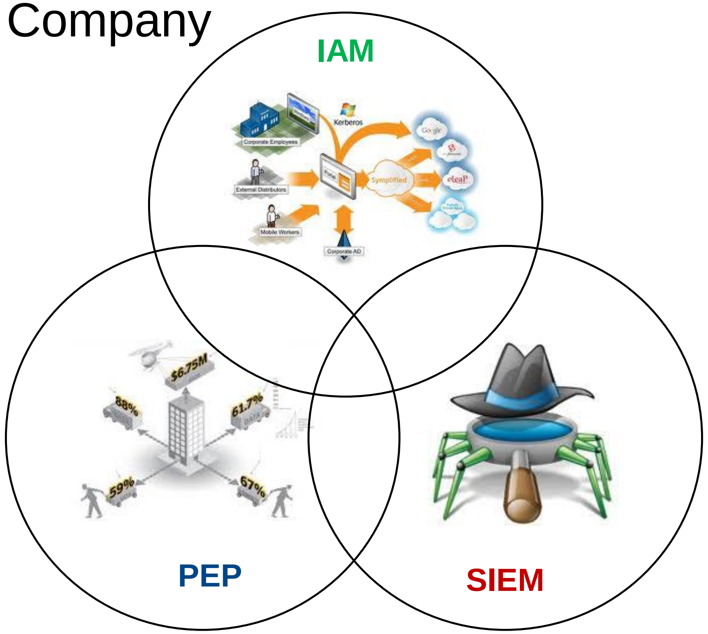


Technology: tools used to perform processes by people

G.1a App Environment: Enterprise Infrastructure

IAM, PEP, SIEM

IAM: management of identity, and its attributes, mapped in virtual tree fashion, according to Company hierarchy



SIEM: management of security information (log) and events (coming from application)

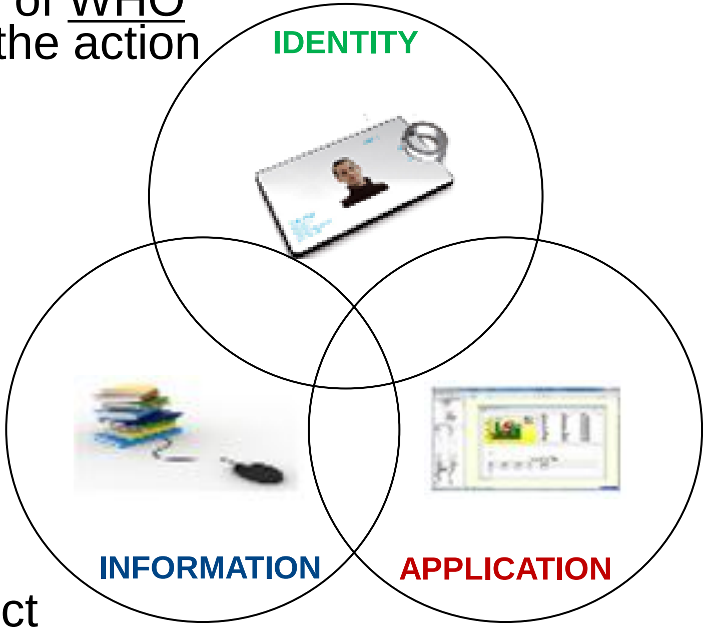
PEP: protection from uncontrolled accesses and data leakage, performed by information users

C4: from Military Environment (Computer/Communication Command & Control)

G.1a App Environment: Models

Identity, Information & Application

Identity: personification into the IT world. Instance of WHO and WHY to perform the action



Application: object composing the IT world. Instantiation of WHEN and WHERE to perform the action

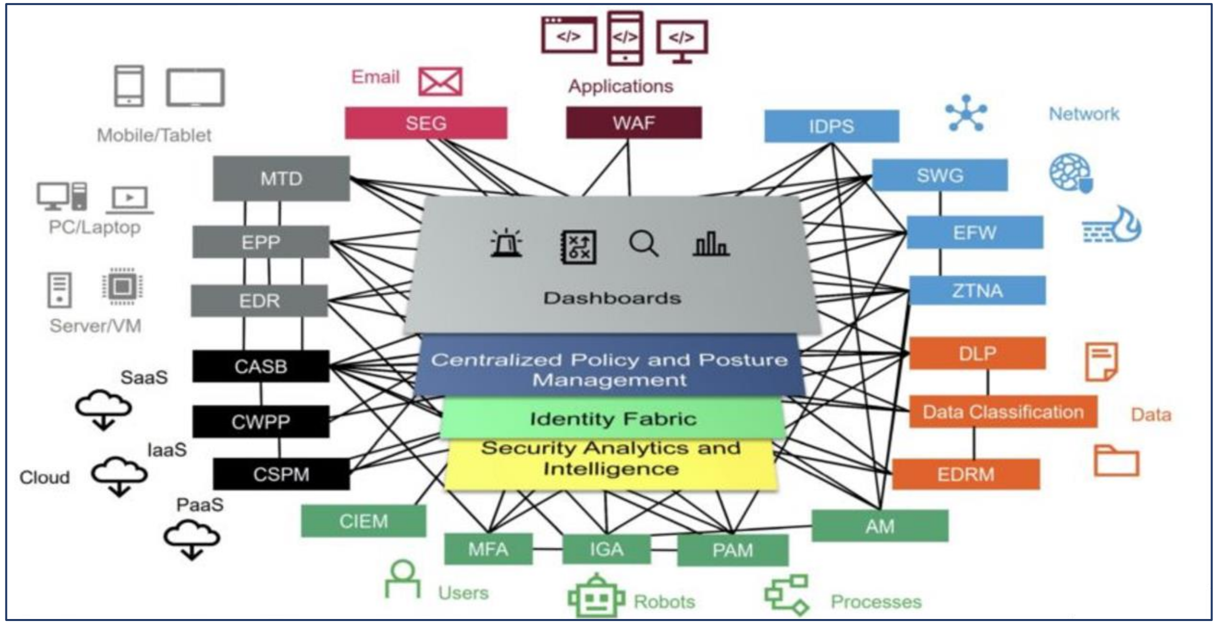
Information: object surrounding the IT world. Instance of WHAT and HOW to perform the action

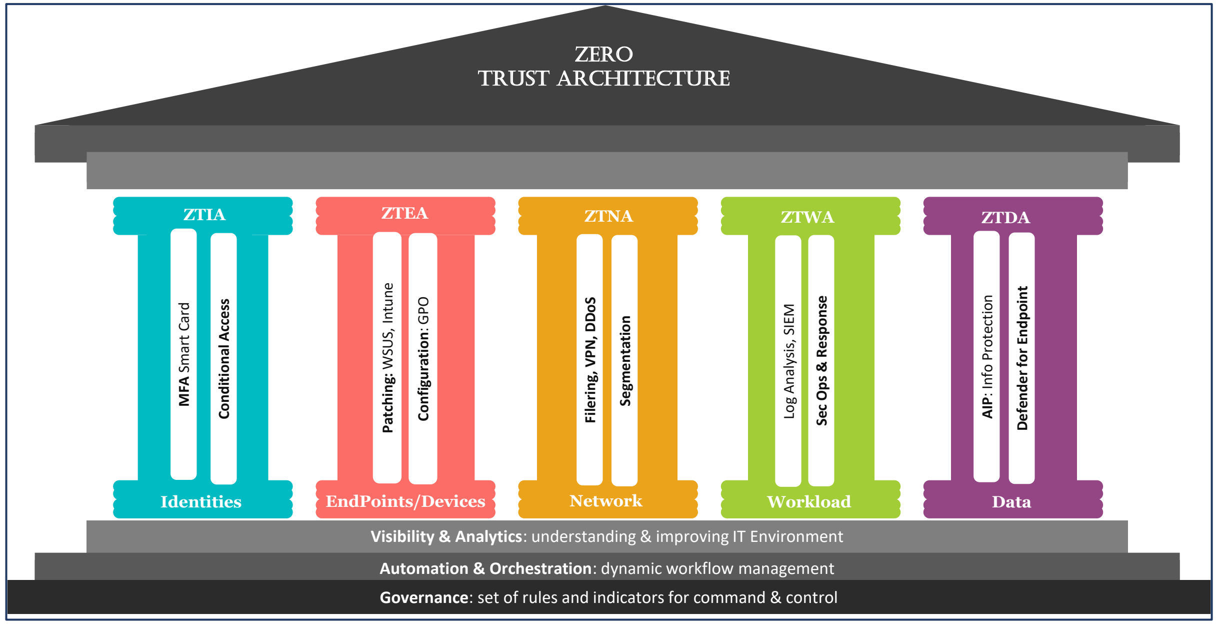
G.1a App Environment

CSMA and ZTA Pillars

CSMA: (ideal) infrastructure for securely operating applications and providing cyber service.

ZTA Pillars: (common) areas of intervention about “never trust, always verify” principle.

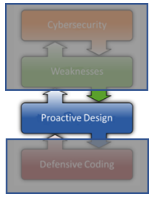




Identity – Endpoint – Network – Workload - Data

G.1a App Environment: Identity

Identity & Access Management and its derivatives

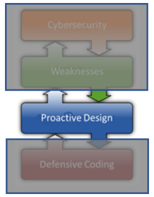


ZTA Pillar	CSMA Tool	Name	Enforce	Enabling
Identity	**IGA**	Identity Governance (SoD)	Authorizations: Permissions	Identity Lifecycle.
Identity	**CIEM**	Cloud Infrastructure Entitlement Management	Roles: Entitlements	Business & Application Lifecycle
Identity	**PAM**	Privileged Access Management	Authorizations: Privileged	Privilege Administration
Identity	**AM**	Access Management	Identity & Access	Proper controls over the access to application functionalities and data
GOV	**IAM**	Identity Fabric	Directory Service: User Attributes	Identity Lifecycle. Role Definition

ZTIA: tools for implementing **AAA**. Mainly focusing on IdM, AM, PAM

G.1a1 App Environment: Identity

IAM – General Concepts: Enabled Features



Seamless Integration: enable business processes, people, and heterogeneous applications to work together seamlessly and securely

Secure Control: Being able to view and control who has access to what resources (**Entitlements**): effectively protecting sensitive information

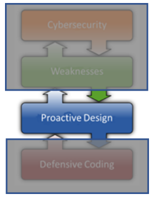
Improved User Experience: transparent and uninterrupted user's experience in interacting with multiple entities in multiple ways

Cost Reduction: reduce administrative costs by automating manual processes.

- **Integrated and integratable solutions across traditional business boundaries**
- **Enabling Compliance with legislative mandates and regulatory requirements (Security, Privacy, and Governance)**
- **Directory-based identity management**
- **Reduce spiraling help desk and other support expenses through capabilities such as automation, self-service, and delegation**

G.1a2 App Environment: Identity

IAM – General Concepts: Expected Benefits



Need to Know: every user can access only the information needed to perform his job

Segregation of Duties: it is possible to activate different application functions for different **roles and** tasks

Approval Workflow: every modification (advancement, special task, leave, etc) **to each user's attribute should be put under specific approval**

Activity Monitoring: collecting, analyzing and (eventually) forgiving evidences of:

- user (strange) behaviours, like misuse, fraud, error, etc
- Administrator (unusual) actions for changing user attributes

– **Proper Role definition**

➔ **Access Control performed accordingly**

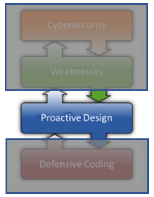
➔ **Compliance to SoD and Privacy legislations**

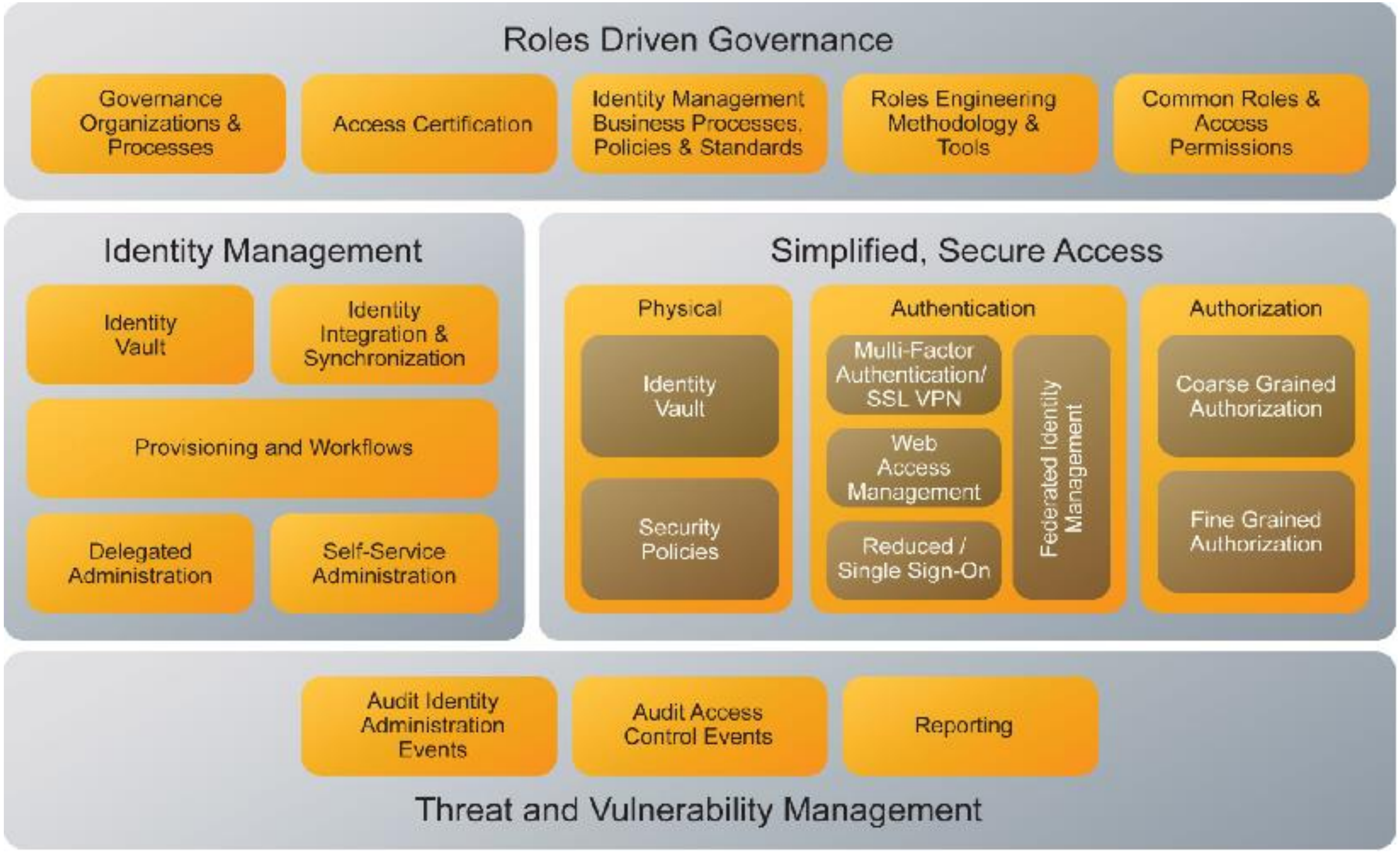
– **System of Validation for every modification, for adhering to official organization change**

– **Infrastructure for collecting Log from systems and applications, in order to analyze information**

G.1a3 App Environment: Identity

IAM – Framework





Roles Driven Governance

Governance Organizations & Processes

Access Certification

Identity Management Business Processes, Policies & Standards

Roles Engineering Methodology & Tools

Common Roles & Access Permissions

Identity Management

Identity Vault

Identity Integration & Synchronization

Provisioning and Workflows

Delegated Administration

Self-Service Administration

Simplified, Secure Access

Physical

Identity Vault

Security Policies

Authentication

Multi-Factor Authentication/ SSL VPN

Web Access Management

Reduced / Single Sign-On

Federated Identity Management

Authorization

Coarse Grained Authorization

Fine Grained Authorization

Audit Identity Administration Events

Audit Access Control Events

Reporting

Threat and Vulnerability Management

Role

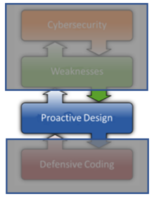
Identity

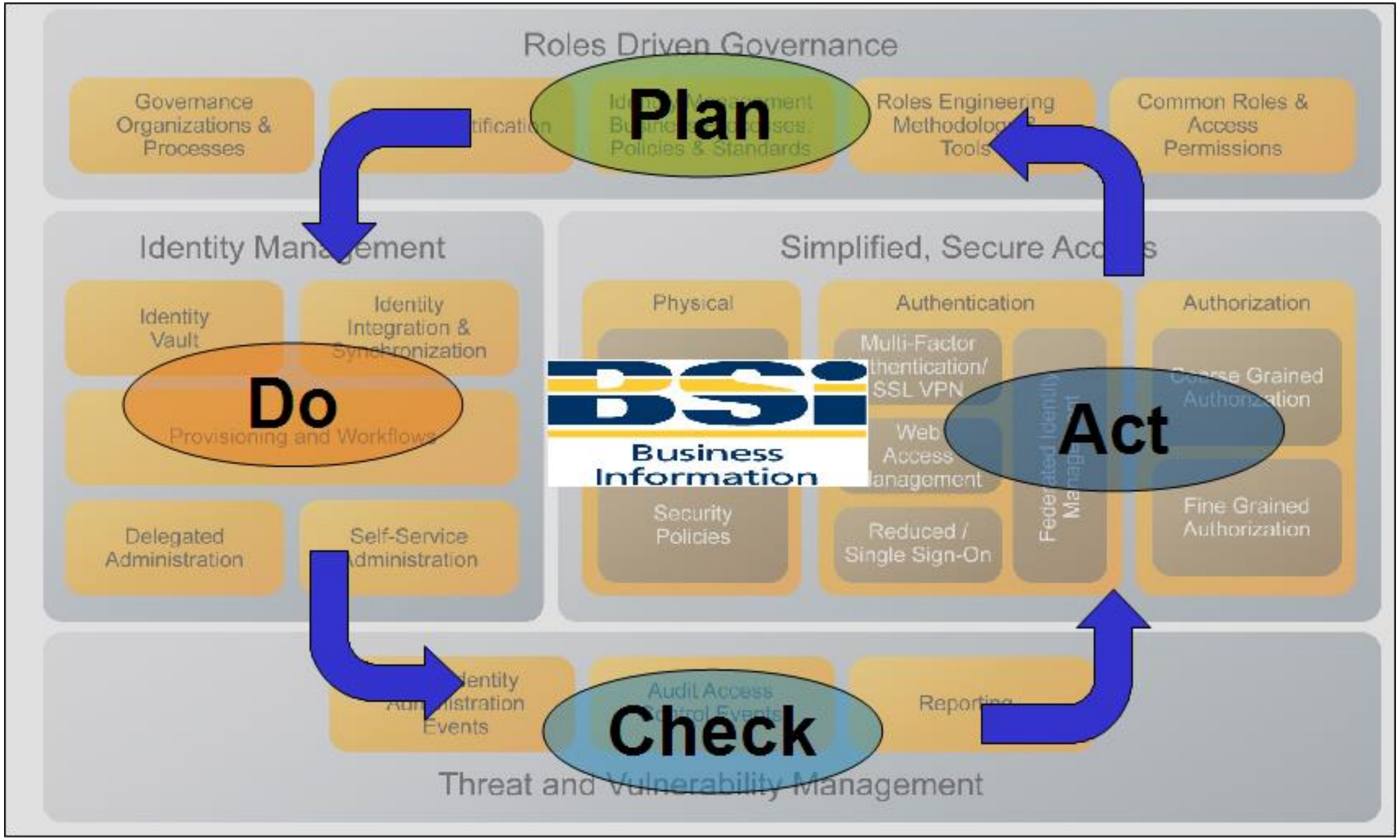
Access

Log

G.1a4 App Environment: Identity

IAM – Framework vs Deming (Shewhart) Cycle



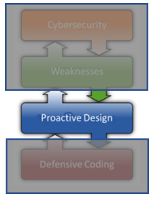

Roles Driven Governance
Governance Organizations & Processes
Certification
Identity Management Business Processes Policies & Standards
Roles Engineering Methodology Tools
Common Roles & Access Permissions
Plan
Identity Management
Identity Vault
Identity Integration & Synchronization
Provisioning and Workflows
Do
Delegated Administration
Self-Service Administration
Simplified, Secure Access
Physical
Security Policies
Authentication
Multi-Factor Authentication/ SSL VPN
Web Access Management
Reduced / Single Sign-On
Federated Identity Management
Authorization
Coarse Grained Authorization
Fine Grained Authorization
Act
BSi
Business Information
Identity Administration Events
Audit Access Control Events
Reporting
Check
Threat and Vulnerability Management

Plan: Role

Do: Identity

Check: Log

Act: Access

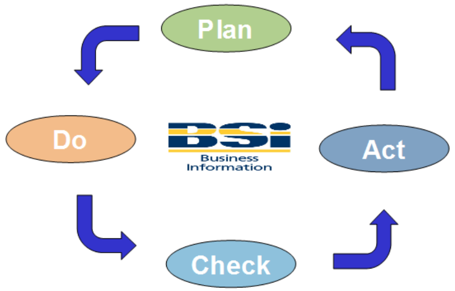


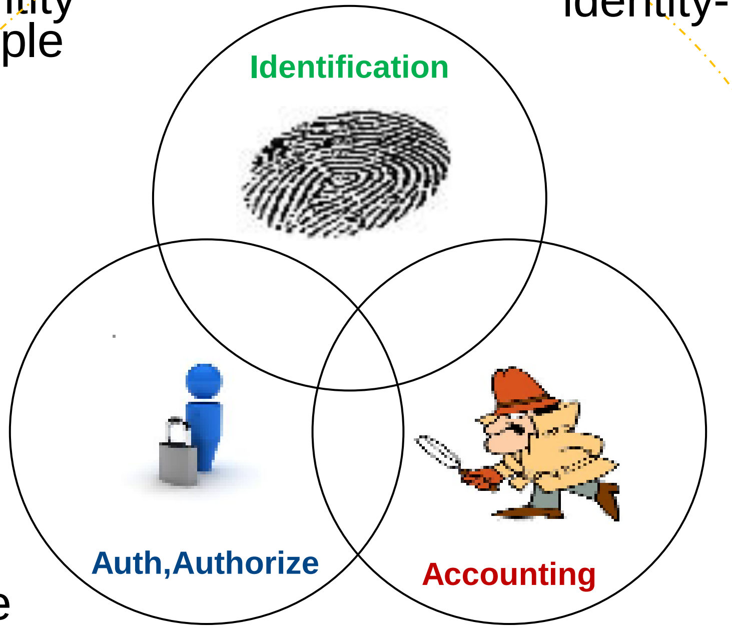
G.1a5 App Environment: Identity

Enabled Functions: Role, Identification, Authorization, Accounting

Identification: personification in the IT world. Virtual Identity association to each people

Governance: plan, enforce, control, clean-up and certify identity-related issues



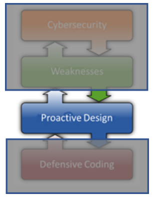


Auth, Authorize: provide proper access to resources and Information on the basis of Attribute values

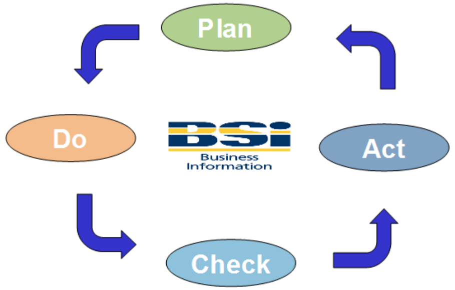
Accounting: monitoring and tracking of accesses and performed operation and accessed data

G.1a6 App Environment: Identity

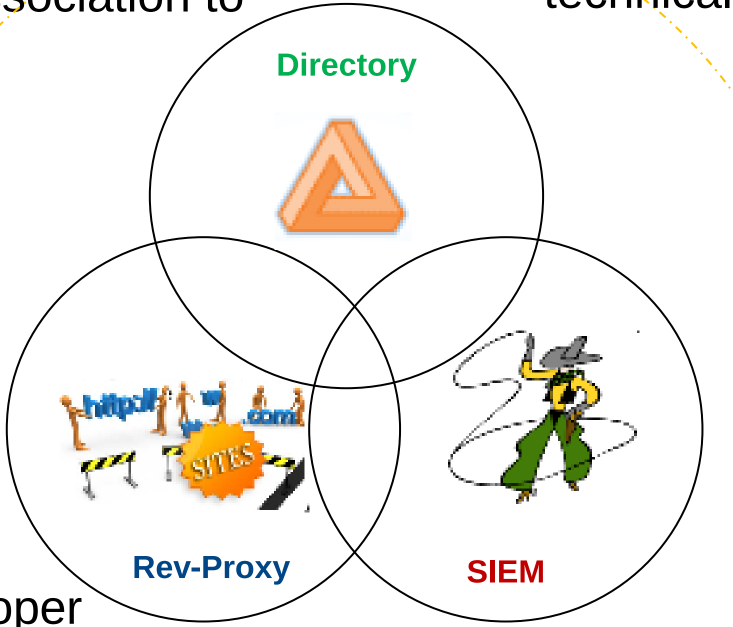
Needed Components: Directory, Rev-Proxy, SIEM



WKF & Engine: core component for activating all human processes and technical interaction



Directory: personification in the IT world. Virtual Identity association to each people

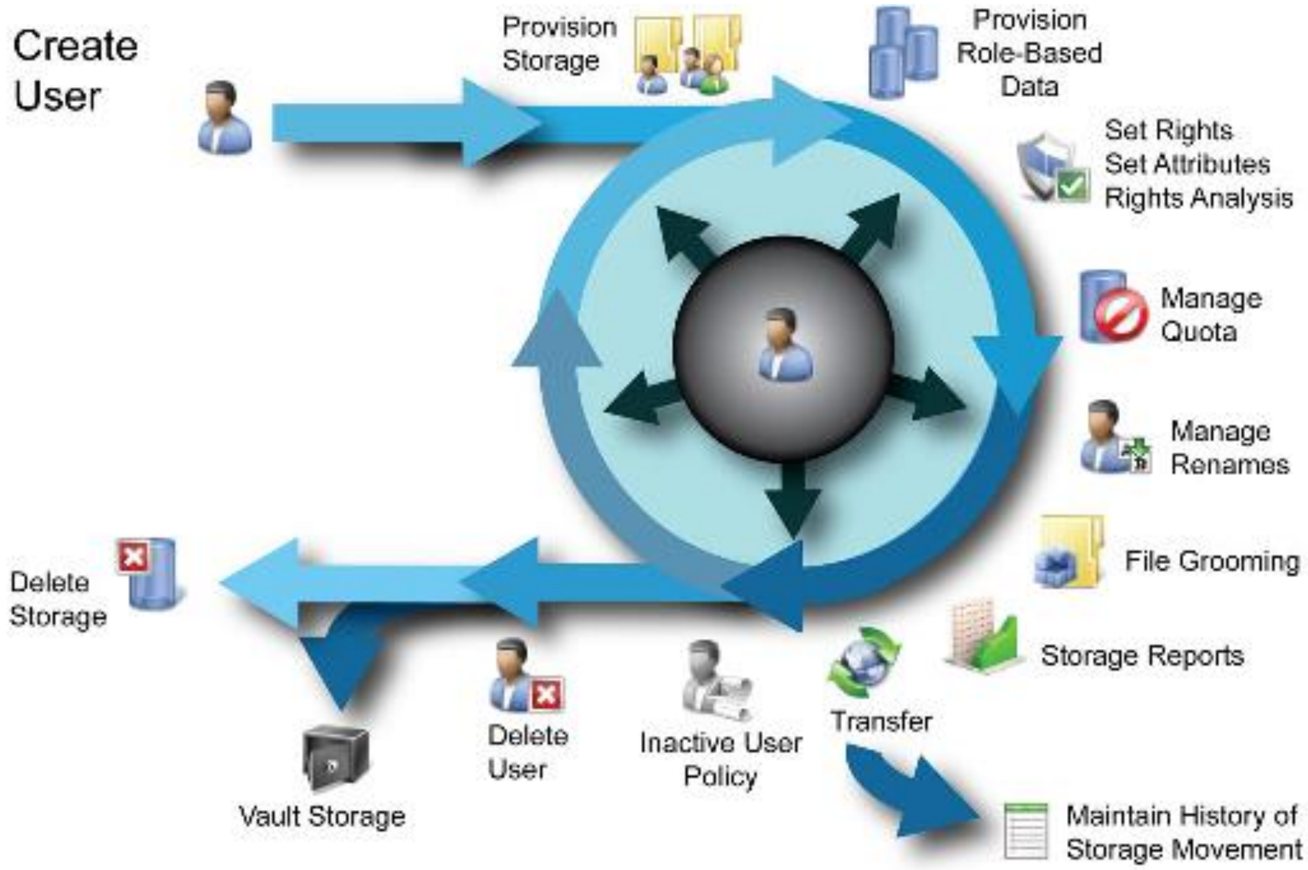


SIEM: monitoring and tracking of accesses and performed operation and accessed data

Rev-Proxy: provide proper access to resources and Information on the basis of Attribute values

G.1a7 App Environment: Identity

Implementation



Identity Life-Cycle

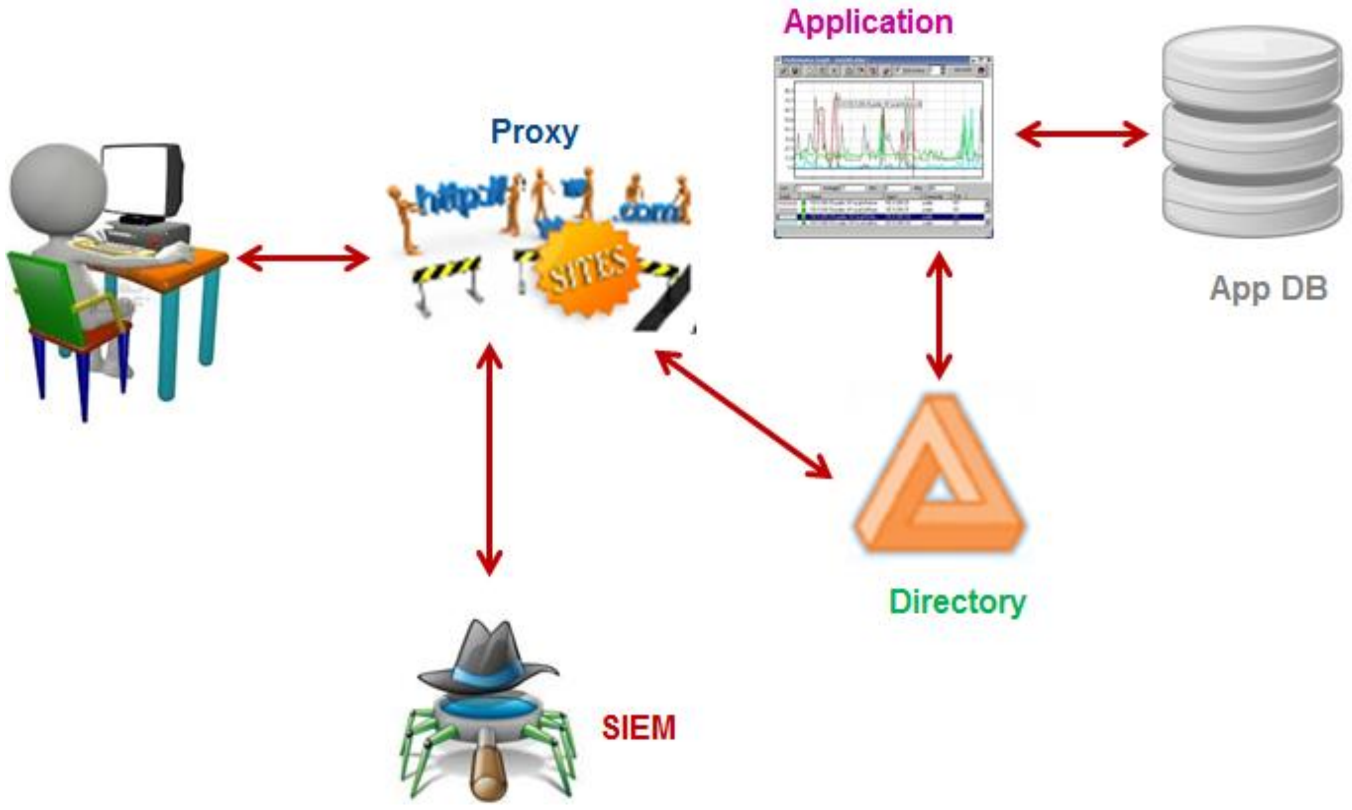
- Single **Entry Point**
- Complete **Census** of identities
- Workflow triggered by Events
- Mapping between **Applications** and **Organization** chart
- Definition of **Roles** (usually as **Entitlements** set)

G.1a8 App Environment: Identity

Implementation

Applications

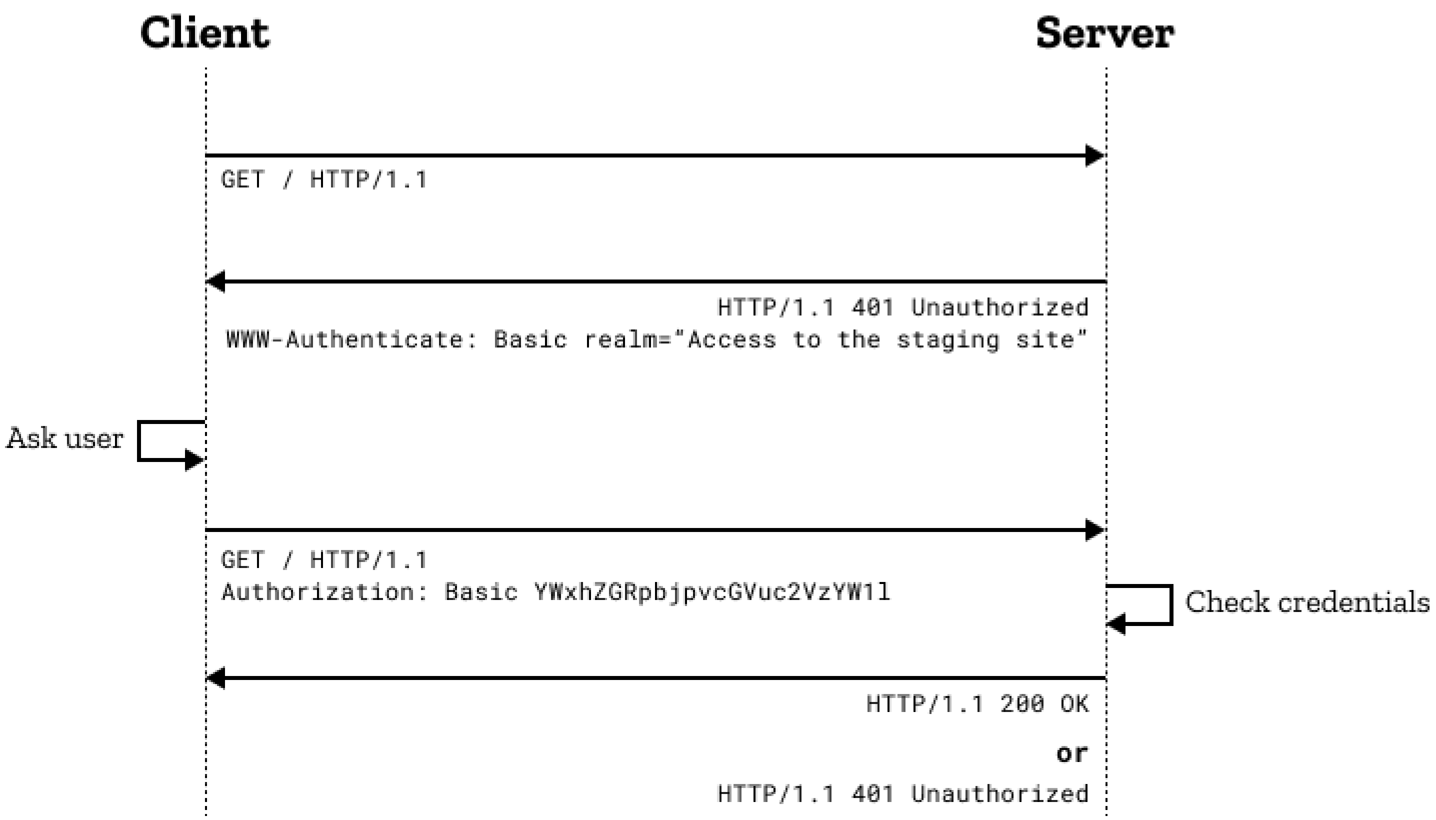
- Identity-ready (Attribute-driven access)
- Home-Made Systems /Client-Server (no HTTP)
- **HTTP Authentcation**
- **Autorization Header**
- IBM Legacy (specific logic)
- Cloud Architecture
- Federation (delegation of responsibility)
- Fine-Grained (XACML)



G.1a9 App Environment: Identity

HTTP Authentication

HTTP provides a general framework for access control and authentication, RFC 7235 defines the **HTTP authentication framework**, which can be used by a server to challenge a client request, and by a client to provide authentication information.



```
GET / HTTP/1.1
```

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Basic realm="Access to the staging site"
```

```
GET / HTTP/1.1
Authorization: Basic YWxhZGRpbjpvcGVuc2VzYW1l
```

```
HTTP/1.1 200 OK
```

```
HTTP/1.1 401 Unauthorized
```

The challenge and response flow works like this:

1. The server responds to a client with a 401 (Unauthorized) response status and provides information on how to authorize with a WWW-Authenticate response header containing at least one challenge.

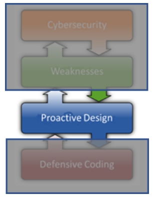
2. A client that wants to authenticate itself with the server can then do so by including an Authorization request header with the credentials.

3. Usually a client will present a password prompt to the user and will then issue the request including the correct **Authorization header**.

See https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication

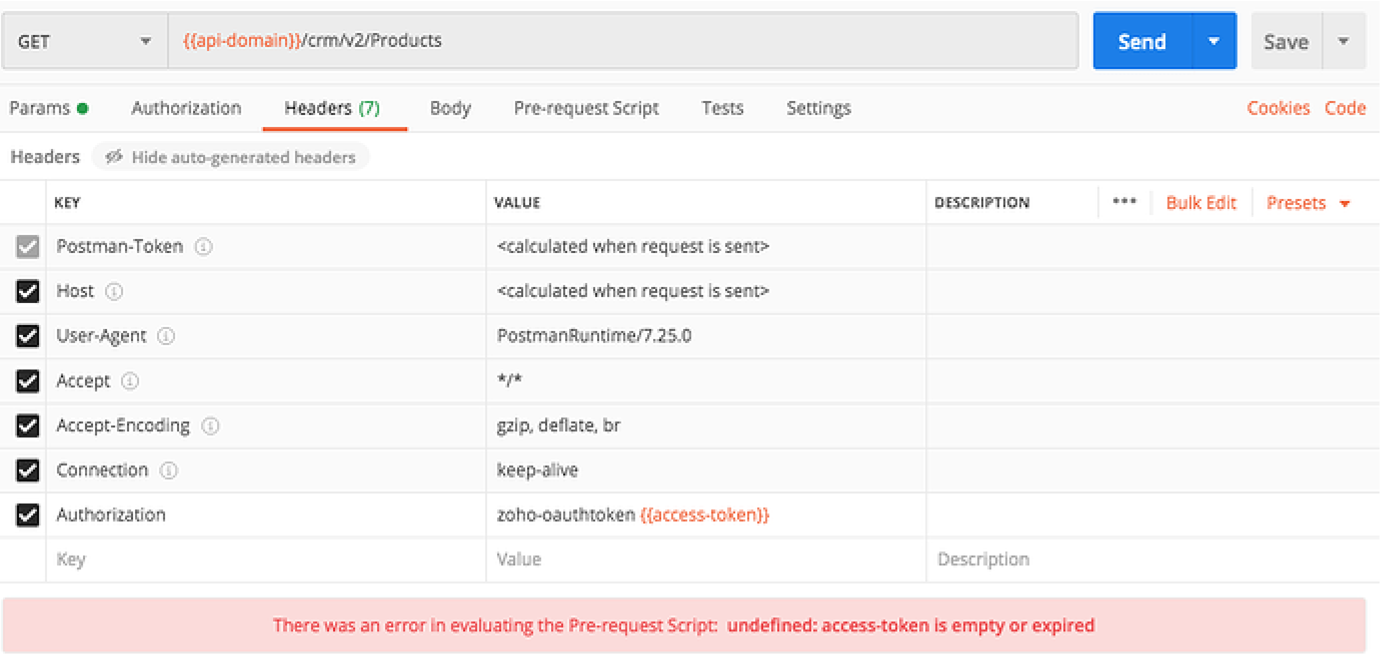
G.1a10 App Environment: Identity

Authorization Header



The HTTP **Authorization request header** can be used to provide credentials that authenticate a user agent with a server, allowing access to a protected resource. It is usually, but not always, sent after the user agent first attempts to request a protected resource without credentials. Syntax:

```
Authorization: <auth-scheme> <authorization-parameters>
```



{{api-domain}}/crm/v2/Products

	KEY	VALUE	DESCRIPTION
☑	Postman-Token	<calculated when request is sent>	
☑	Host	<calculated when request is sent>	
☑	User-Agent	PostmanRuntime/7.25.0	
☑	Accept	*/*	
☑	Accept-Encoding	gzip, deflate, br	
☑	Connection	keep-alive	
☑	Authorization	zoho-oauthtoken {{access-token}}	
	Key	Value	Description

There was an error in evaluating the Pre-request Script: undefined: access-token is empty or expired

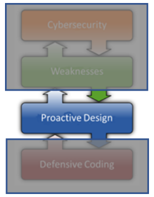
Example:

```
Authorization: Digest
username=<username>,
    realm="<realm>",
    uri="<url>",
    algorithm=<algorithm>,
    nonce="<nonce>",
    nc=<nc>,
    cnonce="<cnonce>",
    qop=<qop>,
    response="<response>",
    opaque="<opaque>"
```

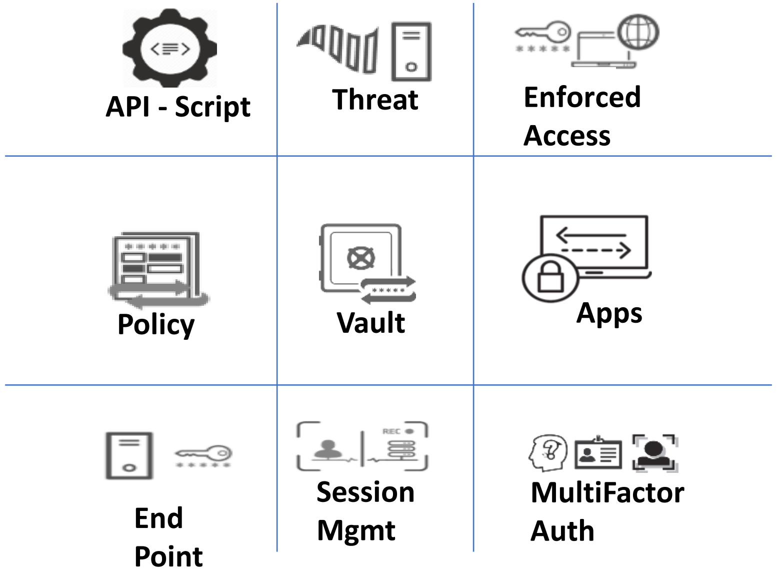
See https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Authorization

G.1a11 App Environment: Privileged Identity

Privileged Access Manager in a nutshell



A Privileged Access Management system is usually composed by 9 logical components



The **Vault** centrally stores the information such as:

- credentials: shared between the managed servers
- configurations: management issues, by the means of **Policy** (Manager)
- log / session recording: provided by **Session Mgmt**

The system administrators access by the **Enforced Access** that allows also for:

- **Multifactor Authentication**
- information intelligence (**Threat**) linkage

The Machine-to-Machine (A2A) access could be automatized by the modules:

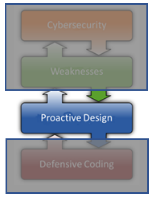
- **API-Script** (system to system)
- **Apps** (application to application)

Finally, the access to user’s information could be securized by **EndPoint**

The components to be installed should be chosen by the protection needs and the implementation maturity

G.1a12 App Environment: Privileged Identity

Privileged Access Manager



PAM is the solution for protecting the privileged access, in order to protect from lateral movements.
The available functionalities are introduced in by layers, based on the maturity of implementation

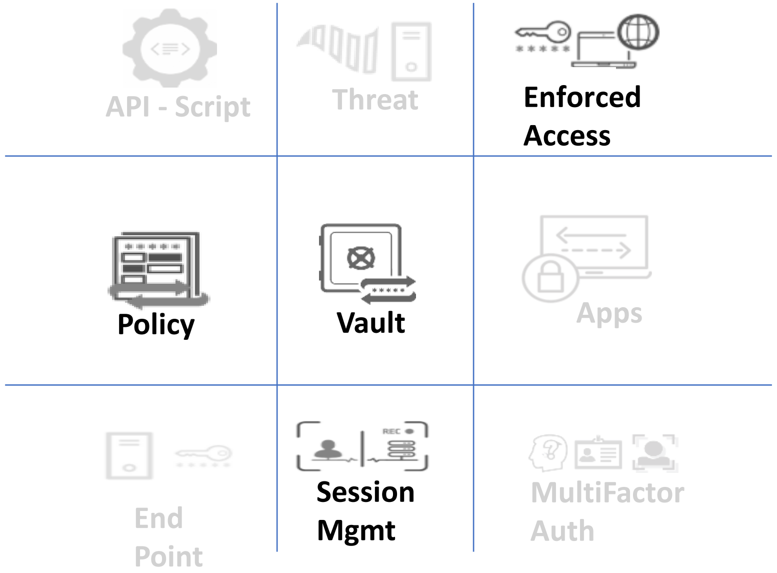
PHASE OF MATURITY

0 NO PAM	1 FOUNDATIONAL	2 ENHANCED	3 ADAPTIVE
High Risk	Get **Visibility**	Integrate **policies**	Increase **automation**
	Reducing **attack surface**	limiting **overprivileged** users	Improving **intelligence**
	• Credentials Vault • Password Rotation • Autologon on Target • Breaking Glass Procedure • SIEM Integration • Centralized Access	• A2A Password mgmt • MFA • Logic Segmentation • Privilege Elevation • Delegation mgmt • Session Recording • File mgmt	• SSO • Workflows • Command white/black list • EUBA – Risk Based Approval • Audit Role and Compliance • VPN-Less access

G.1a11 App Environment: Privileged Identity

Privileged Access Manager – Core Security Components

The Core Privileged Account Security components



The **Vault** centrally stores the information such as:

- credentials: shared between the managed servers
- configurations: management issues, by the means of **Policy** (Manager)
- log / session recording: provided by **Session Mgmt**

The system administrators access by the **Enforced Access** that allows also for:

- **Multifactor Authentication**
- information intelligence (**Threat**) linkage

The Machine-to-Machine (A2A) access could be automatized by the modules:

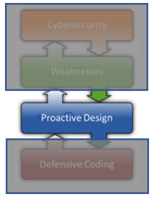
- **API-Script** (system to system)
- **Apps** (application to application)

Finally, the access to user’s information could be securized by **EndPoint**

The components to be installed should be chosen by the protection needs and the implementation maturity

G.1b App Environment: Endpoint

Protection from Threats targeting Endpoints



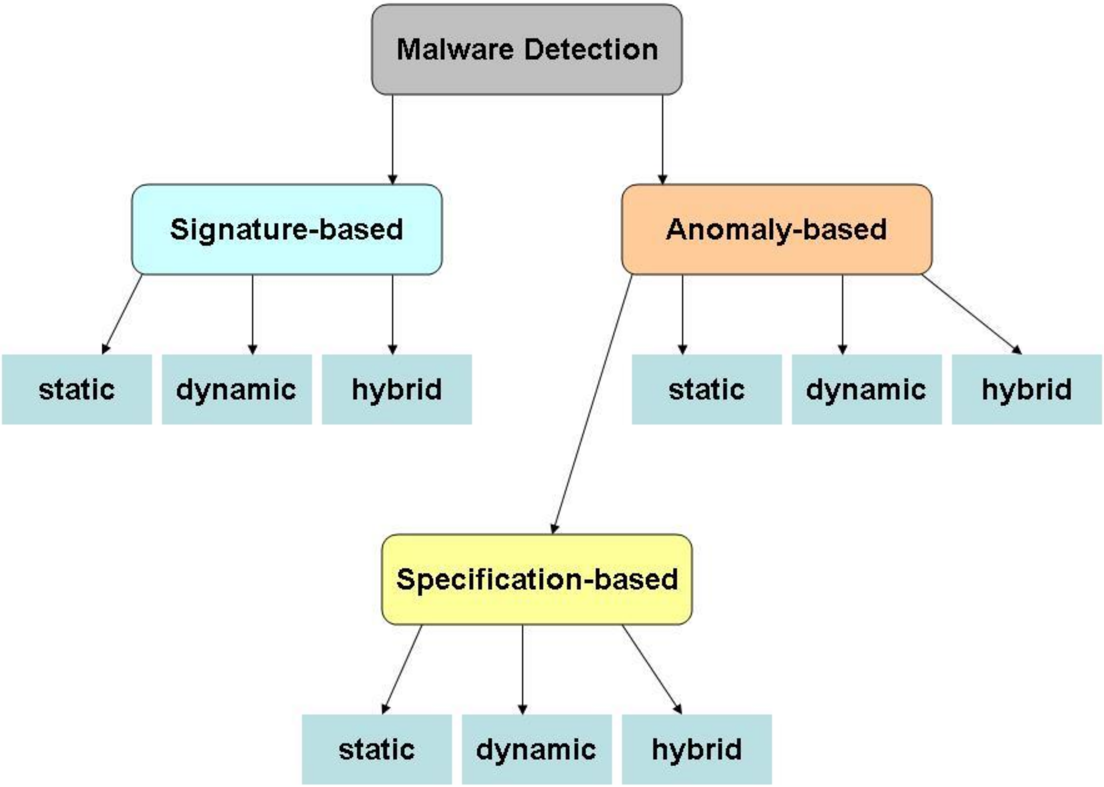
ZTA Pillar	CSMA Tool	Name	Enforce	Enabling
EndPoint	CMDB	Asset Mgmt	Item identification	Item Configuration
EndPoint	MDM	Mobile Device Management	Patching	Vulnerability Management; Change & Configuration Mgmt
EndPoint	ATP MTD EPP EDR	Advanced Threat Prevention Mobile Threat Detection EndPoint Protection Endpoint Detection and Response	Blocking threats	Spreading across endpoints and nets.
GOV	PEP	Policy Enforcement Point	Centralized Policy	Posture Management

ZTPA: tools for implementing the **protection** of endpoint. Mainly focusing on ATP

G.1b1 App Environment: Endpoint

Advanced Threat Protection

ATP: category of security solutions that defend against sophisticated malware or hacking-based. Three basic techniques:



Deviation from usual (eventual “deflagration” in a sandbox)

➔ **Anomaly**: **User Behavior Analysis** (e.g. unusual System Calls) Useful for detecting **Zero-Day**. Anomaly-based Dynamic is the only way for addressing **File-less**,

➔ **Specification**: recognition of previously described patterns of compromission, using generally available information, like **IoC** (Indicator of Compromission) and **IoA** (Indicator of Attack).

Signature: a specific pattern (sequence of bites) that allows cybersecurity technologies to recognize malicious threats that has already been:

- discovered in the wild
- cataloged as part of a database
- Remained unchanged (**not for nowadays polymorphic malware**)

(Viruses are only for MS-DOS systems, since require to be inserted inside executable files, see http://www.di-srv.unisa.it/~ads/corso-security/www/HTML/VIRUS/node26.html)

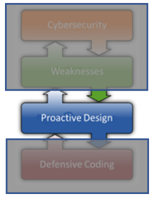
➔ Dynamic and Hybrid Analysis

1. requires additional elaboration effort.
2. may generate false-positive, for “strange” tasks performed by the application.

See https://profsandhu.com/cs5323_s17/im_2007.pdf

G.1c App Environment: Network

Filtering unwanted traffic



ZTA Pillar(s)	CSMA Tool	Name	Enforce	Enabling
Network	CNS IDPS	Cloud Network Security Intrusione Detection/Prevention System	Segregation & Segmentation	Micro-Segmentation
Network	DDoS	Anti-DDoS	Protect against obscuration	Application Availability
Network	DNS-Sec	DNS Security	predicting, blocking, and tracking malicious activity against name resolution	Access to Internet
Network	SWG	Secure Web Gateway	URL Filtering	Access to Internet
Network	VPN	Virtual Private Network	threats, and data leakage	Access to shadow IT
Network	SD-WAN	SW Defined WAN	intelligent unified view and simplified mgmt	Traffic Prioritization, WAN Optimization, converged backbones)
Network	EFW	FW as a Service Enterprise Firewall	Next Generation Rules	Net Filtering
GOV	SOAR	Security Orchestration Automation and Response	Security Analytics	Fast Response

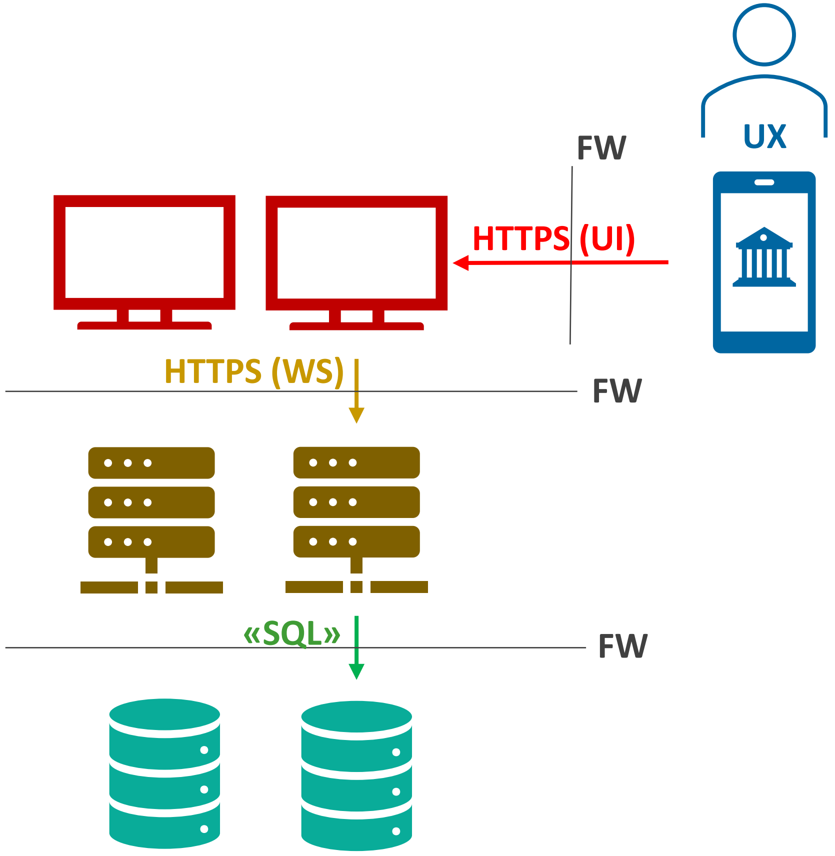
ZTNA: tools for implementing network **filters** and **log** management. Mainly focusing on network layers (tiers).

G.1c1 App Environment: Network

Three Tiers: Splitting the Application into layers

Distributed elaboration: **Three Tier**: system architecture based on the segregation into 3 layers, each having its peculiarities:

1. **Presentation** (Web Server),
2. **Elaboration** (App Server),
3. **Data** (DB Server).



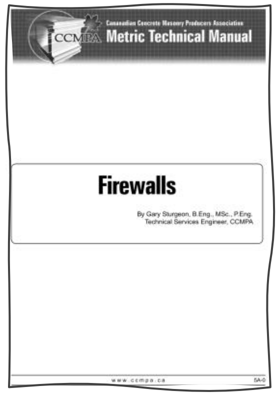
Each one on different item instance, hosts and networks (separated by Firewalls)

G.1c2 App Environment: Network

Firewall: historical background

The concept comes from masonry: «A firewall is the ultimate defense against the spread of fire. It must be able to withstand the onslaught of a fire and prevent further fire spread by containing it to one side of the wall until the fire burns itself out, or is extinguished.”

(see https://ccmpa.ca/wp-content/uploads/2012/02/Final_2013Sec5A.pdf)

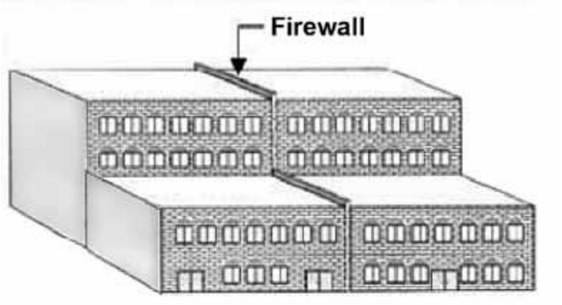


Canadian Concrete Masonry Producers Association
CCMPA Metric Technical Manual

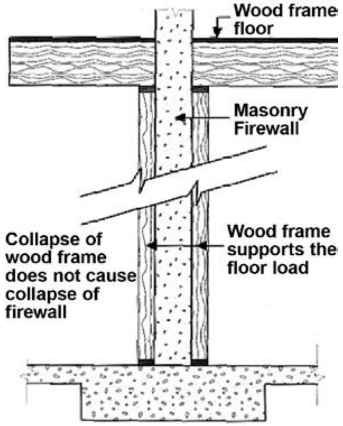
Firewalls

By Gary Sturgeon, B.Eng., MSc., P.Eng.
Technical Services Engineer, CCMPA

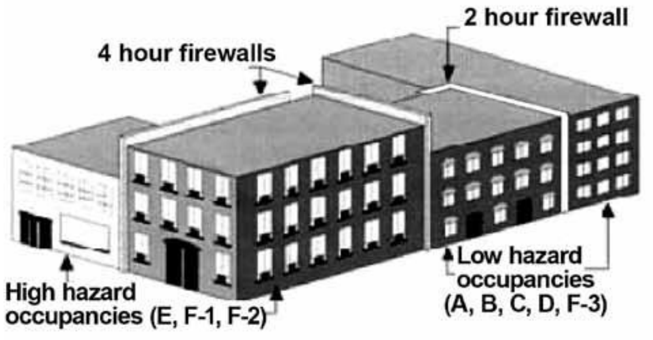
www.ccmpa.ca
5A-1



Separation of Buildings


Wood frame floor
Masonry Firewall
Collapse of wood frame does not cause collapse of firewall
Wood frame supports the floor load

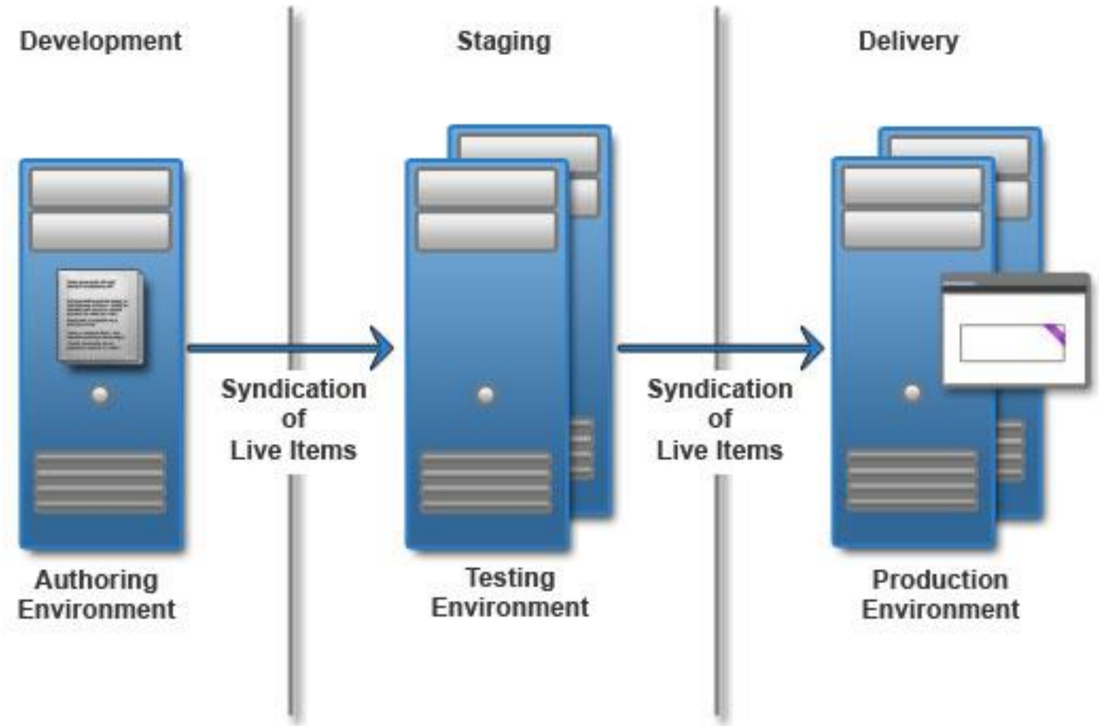
Structural Integrity: Where a floor or roof member is framed into a firewall, the remaining masonry must have sufficient equivalent thickness to provide the fire-resistance rating required by the firewall.


2 hour firewall
4 hour firewalls
High hazard occupancies (E, F-1, F-2)
Low hazard occupancies (A, B, C, D, F-3)

Separation of Major Occupancies

G.1c3 App Environment: Network

Environments

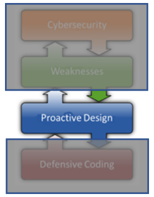


Different implementation and integration of the application

- **DEV** (**Development**): where the application is developed and the changed to the code are applied. In most simple form, it can be a small server or workstation with similar software of same version running.
- **TEST** (Testing): where human check new and changed code via automated or non-automated techniques.,
- **STAGING**: for testing on hardware and software architecture which exactly resembles the production environment
- **LIVE** (**Delivery**, Production): effective application, available to final users .

G.1d App Environment: Workload

Defense from attacks against applications



ZTA Pillar(s)	CSMA Tool	Name	Enforce	Enabling
Workload	SCM	SW Configuration Mgmt	Config & Change	Approval Workflow
Workload	CSPM	Cloud Security Posture Mgmt	Secure Configuration	Compliance
Workload	CWPP	Cloud Workload Protection	SW Mgmt	Configuration Management
Workload	CASB	Cloud Access Security Broker	threats, and data leakage identification	Access to cloud applications and shadow IT
Workload	SEG	Secure Email Gateway	anti-Spamming	anti-Phishing
Workload	WAF	Web Application Firewall	Common Web weakness filtering	Additional layer of defense on published applications
GOV	IRM	Integrated Risk Management	Security Dashboard	Security Governance by KPI

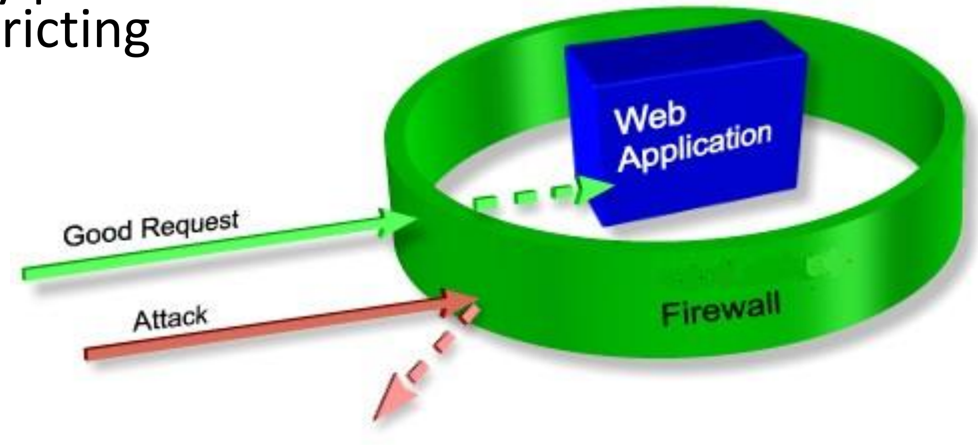
ZTWA: tools for implementing defense from **attacks** against the applications. Mainly focusing on **WAF**, **CWPP**, **CASB**.

G.1d1 App Environment: WAF

IT Sec & Web 2.0: Access, Capability, Network

Access: issuing the correct user to enter the proper application functionality

Capability: instructing proper security profile for programs, restricting capabilities



Network: monitoring, controlling and taking advantage of the proper means to pass information through

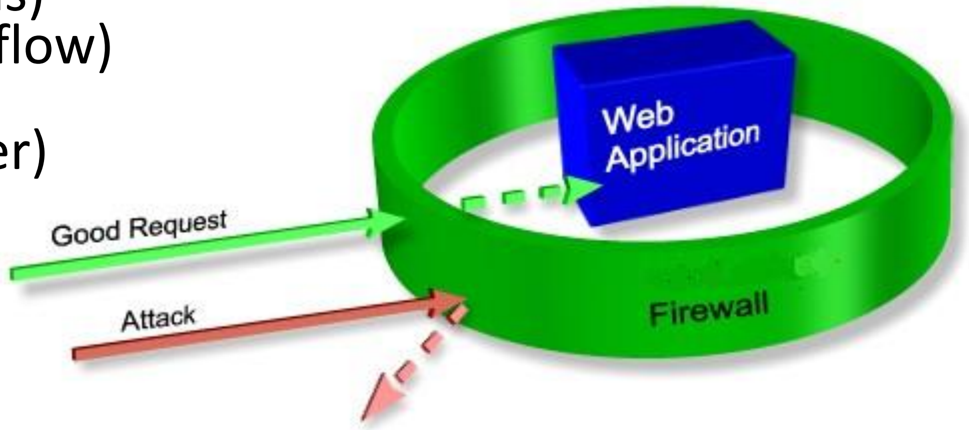
AM implies a GW → IT Command & Control

G.1d2 App Environment: WAF

IT Sec & Web 2.0: Access, Capability, Network

Capability

- **Web Application Firewall** (WebApp)
- App Armor (apps)
- SysTrace (sys calls)
- PaX (buffer overflow)
- RBAC (user)
- GRSecurity (other)



Access

- Identity Server (directory)
- Access Gateway (enforce)
- iManager (admin)

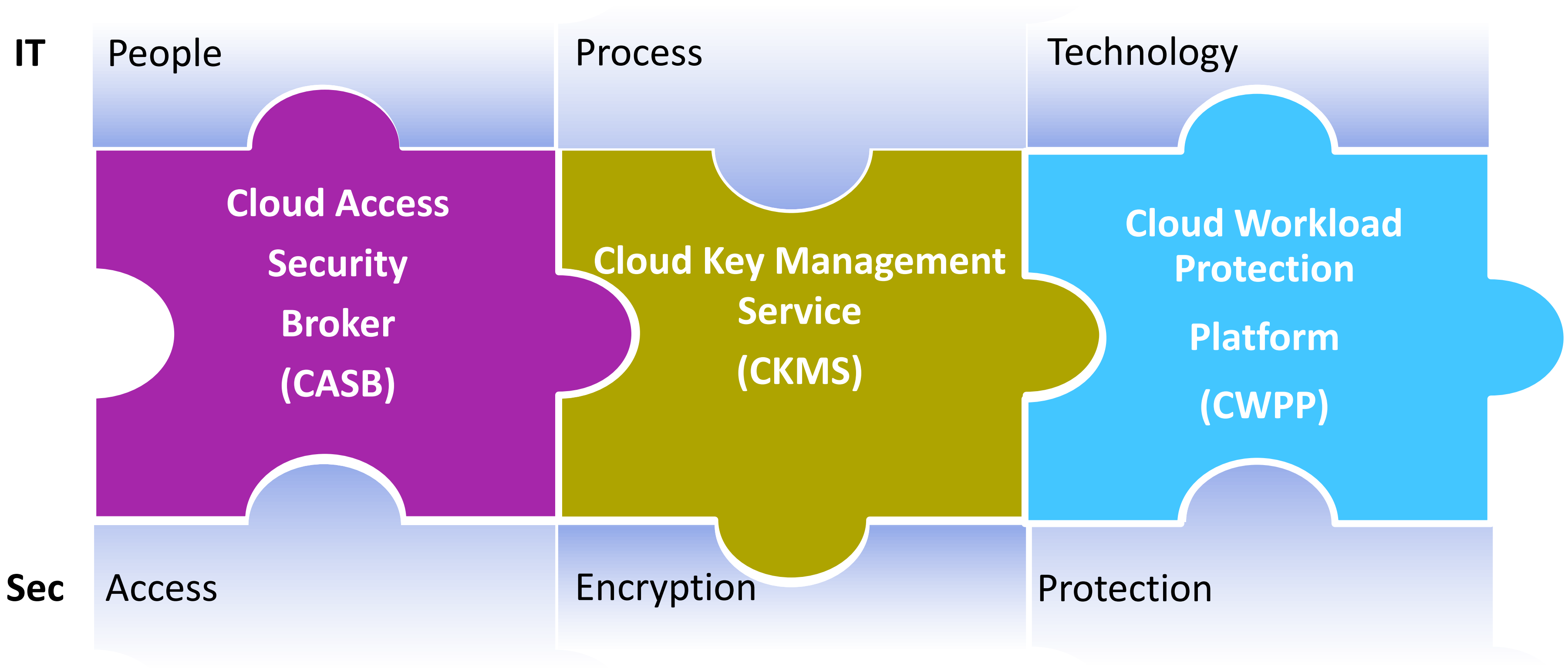
Network

- Iptables (FW)
- Snort (IPS)
- Load Balancing (BC/DR)

AM implies a GW → IT Command & Control

G.1d3 App Environment: Access, Encryption, Protection

CASB – CKMS - CWPP


IT
People
Process
Technology
Cloud Access Security Broker (CASB)
Cloud Key Management Service (CKMS)
Cloud Workload Protection Platform (CWPP)
Sec
Access
Encryption
Protection

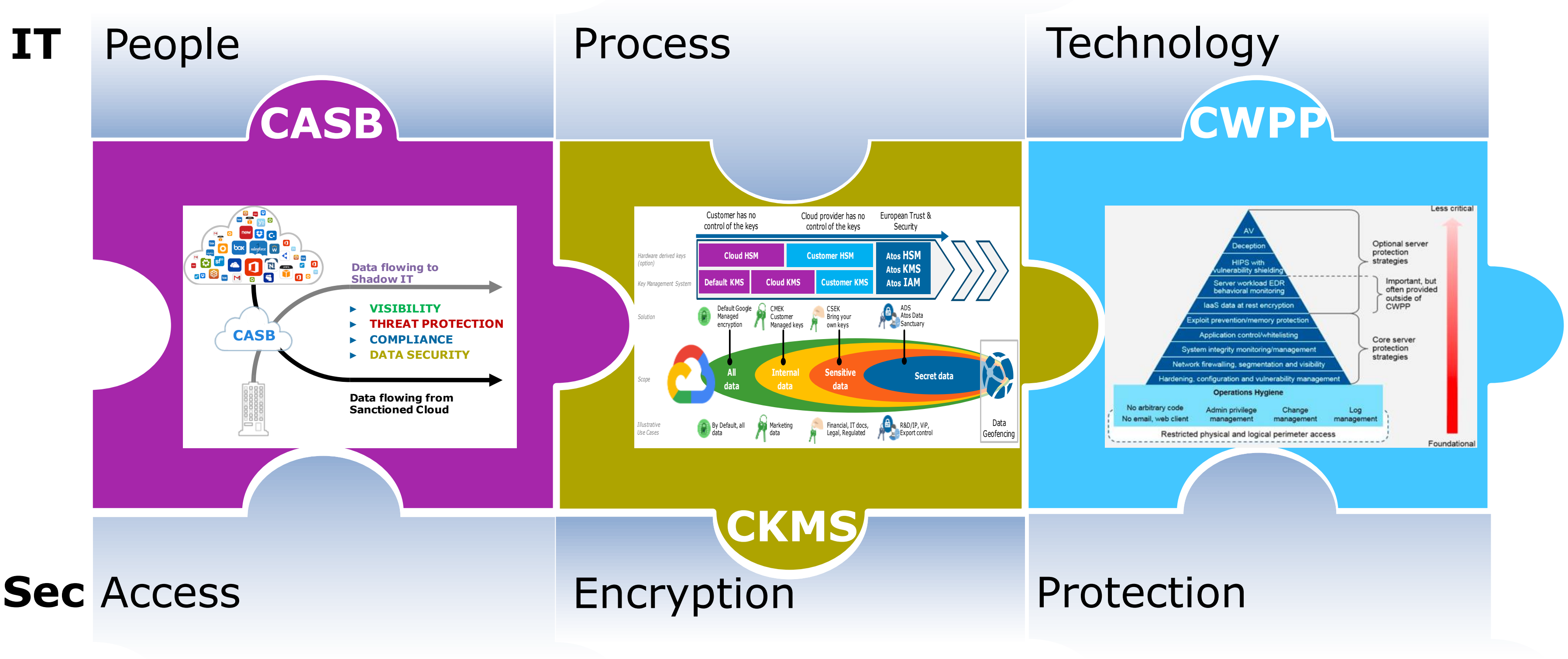
CASB: **Policy Enforcement Point** for all **Communication** between Enterprise and Cloud resources.

CKMS: Key **Management** Solution for **Enforced Cryptography** (described in the next section about Data pillar)

CWPP: Enforcement Point for **securely managing** the **Cloud Workloads** and **Applications**.

G.1d4 App Environment: Access, Encryption, Protection

CASB – CKMS - CWPP


IT
People
Process
Technology
CASB
CKMS
CWPP
Sec
Access
Encryption
Protection
CASB
Data flowing to Shadow IT
VISIBILITY
THREAT PROTECTION
COMPLIANCE
DATA SECURITY
Data flowing from Sanctioned Cloud
Customer has no control of the keys
Cloud provider has no control of the keys
European Trust & Security
Hardware derived keys (option)
Key Management System
Cloud HSM
Customer HSM
Default KMS
Cloud KMS
Customer KMS
Atos HSM
Atos KMS
Atos IAM
Solution
Default Google Managed encryption
CMEK Customer Managed keys
CSEK Bring your own keys
ADS Atos Data Sanctuary
Scope
All data
Internal data
Sensitive data
Secret data
Illustrative Use Cases
By Default, all data
Marketing data
Financial, IT docs, Legal, Regulated
R&D/IP, ViP, Export control
Data Geofencing
AV
Deception
HIPS with vulnerability shielding
Server workload EDR behavioral monitoring
IaaS data at rest encryption
Exploit prevention/memory protection
Application control/whitelisting
System integrity monitoring/management
Network firewalling, segmentation and visibility
Hardening, configuration and vulnerability management
Operations Hygiene
No arbitrary code
No email, web client
Admin privilege management
Change management
Log management
Restricted physical and logical perimeter access
Optional server protection strategies
Important, but often provided outside of CWPP
Core server protection strategies
Less critical
Foundational

CASB: **Policy Enforcement Point** for all **Communication** between Enterprise and Cloud resources.

CKMS: Key **Management** Solution for **Enforced Cryptography** (described in the next section about Data pillar)

CWPP: Enforcement Point for **securely managing** the **Cloud Workloads** and **Applications**.

G.1d5 App Environment: Access

Cloud Access Security Broker

CASB: Security Control Point

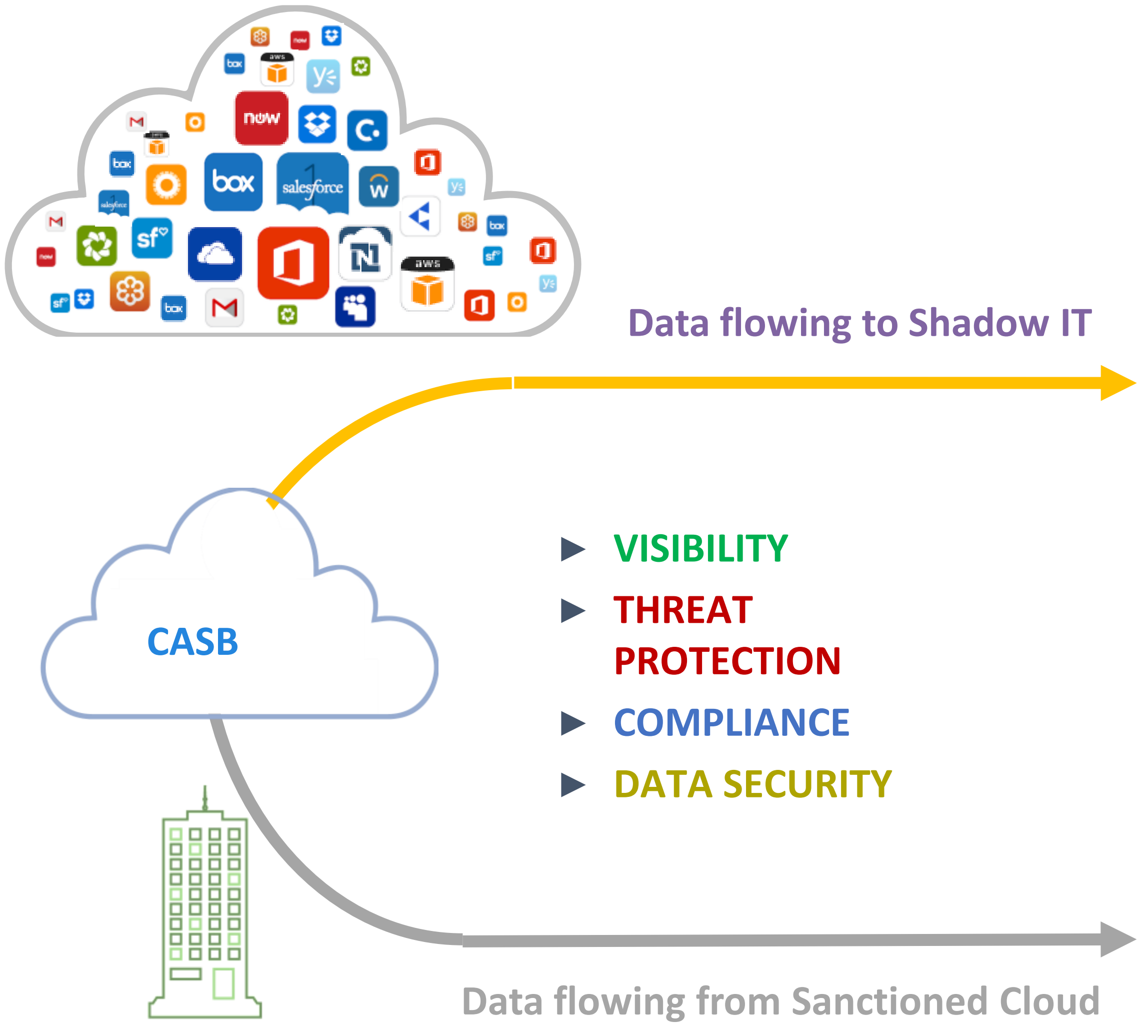
Gartner´s CASB 4 pillars

VISIBILITY: Uncover Shadow IT usage and understand content flowing into and out of the cloud

THREAT PROTECTION: Detect and prevent data exfiltration, insider threat, compromised accounts, and malware

COMPLIANCE: Achieve compliance with both internal policies and industry regulations

DATA SECURITY: Protect data from unauthorized access stemming from a security breach or inadvertent disclosure



Data flowing to Shadow IT

CASB

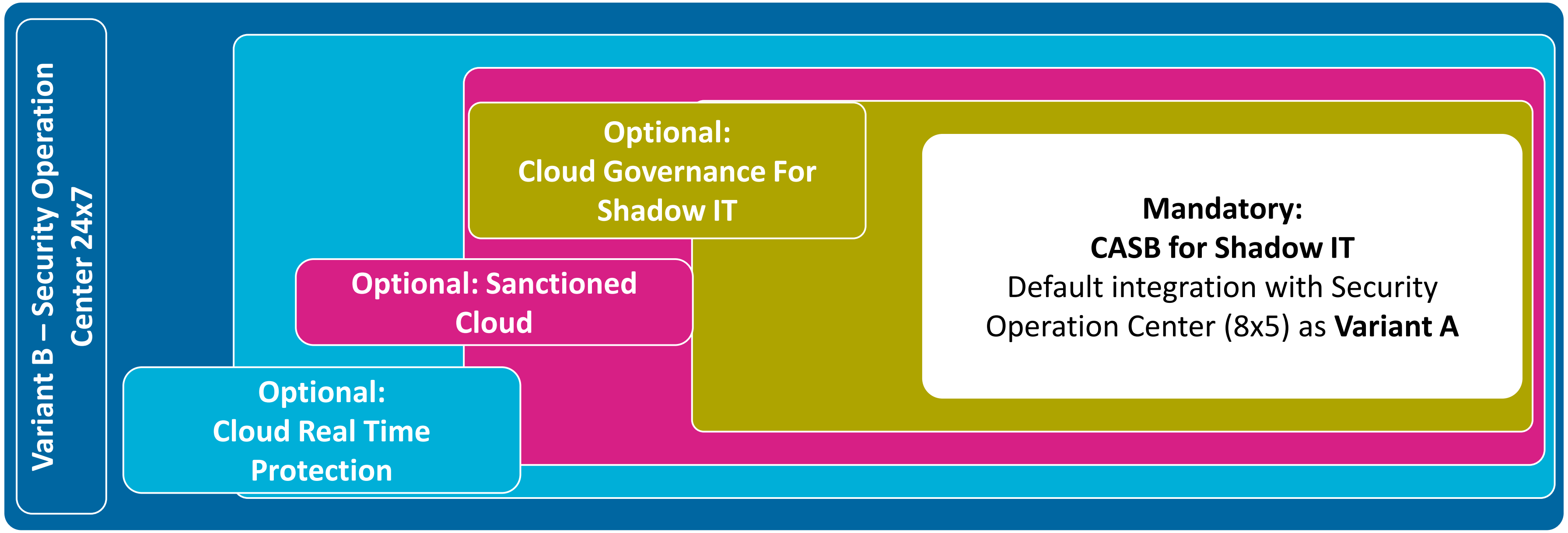
- VISIBILITY
- THREAT PROTECTION
- COMPLIANCE
- DATA SECURITY

Data flowing from Sanctioned Cloud

G.1d6 App Environment: Access

Cloud Access Security Broker

CASB: Logical structure of service modules



Variant B – Security Operation Center 24x7

Optional:
Cloud Real Time Protection

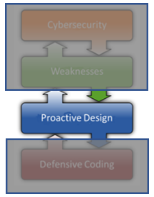
Optional: Sanctioned Cloud

Optional:
Cloud Governance For Shadow IT

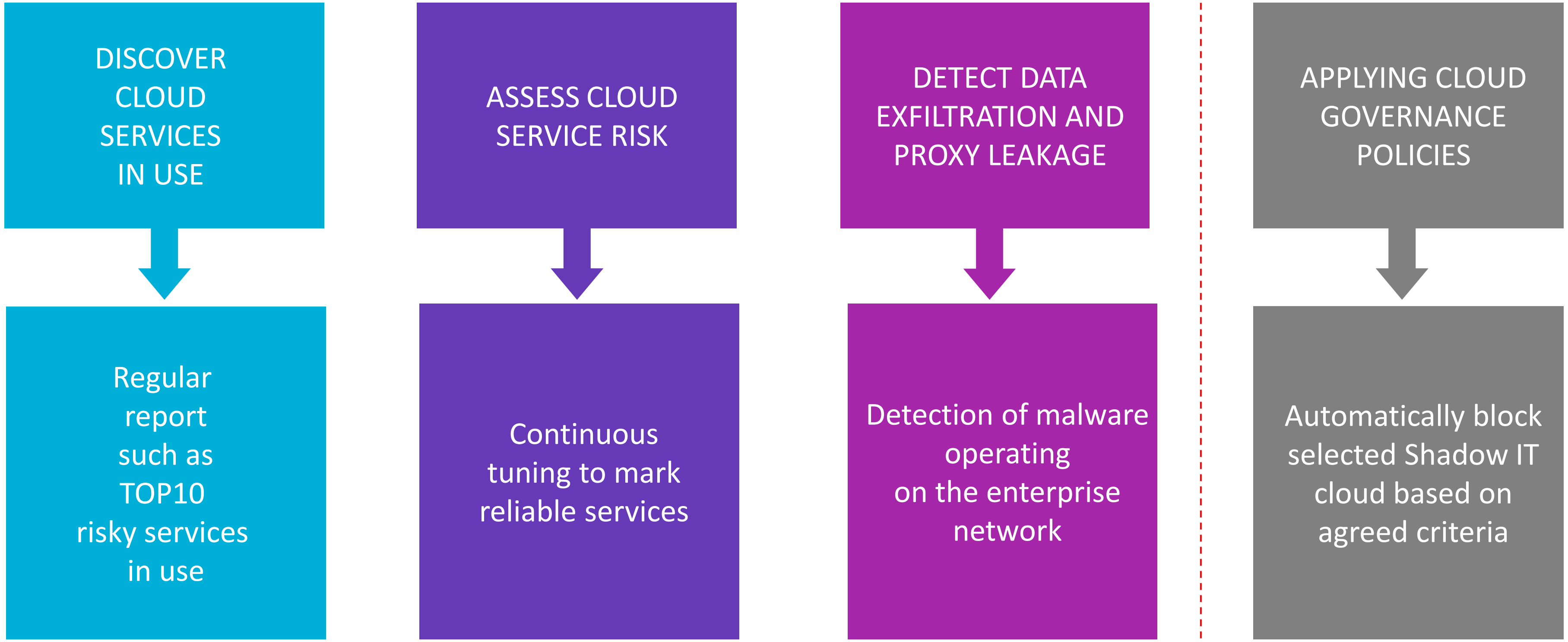
Mandatory:
CASB for Shadow IT
Default integration with Security Operation Center (8x5) as **Variant A**

G.1d7 App Environment: Access

Cloud Access Security Broker



CASB: Shadow IT use cases



DISCOVER CLOUD SERVICES IN USE

Regular report such as TOP10 risky services in use

ASSESS CLOUD SERVICE RISK

Continuous tuning to mark reliable services

DETECT DATA EXFILTRATION AND PROXY LEAKAGE

Detection of malware operating on the enterprise network

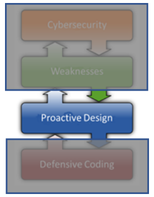
APPLYING CLOUD GOVERNANCE POLICIES

Automatically block selected Shadow IT cloud based on agreed criteria

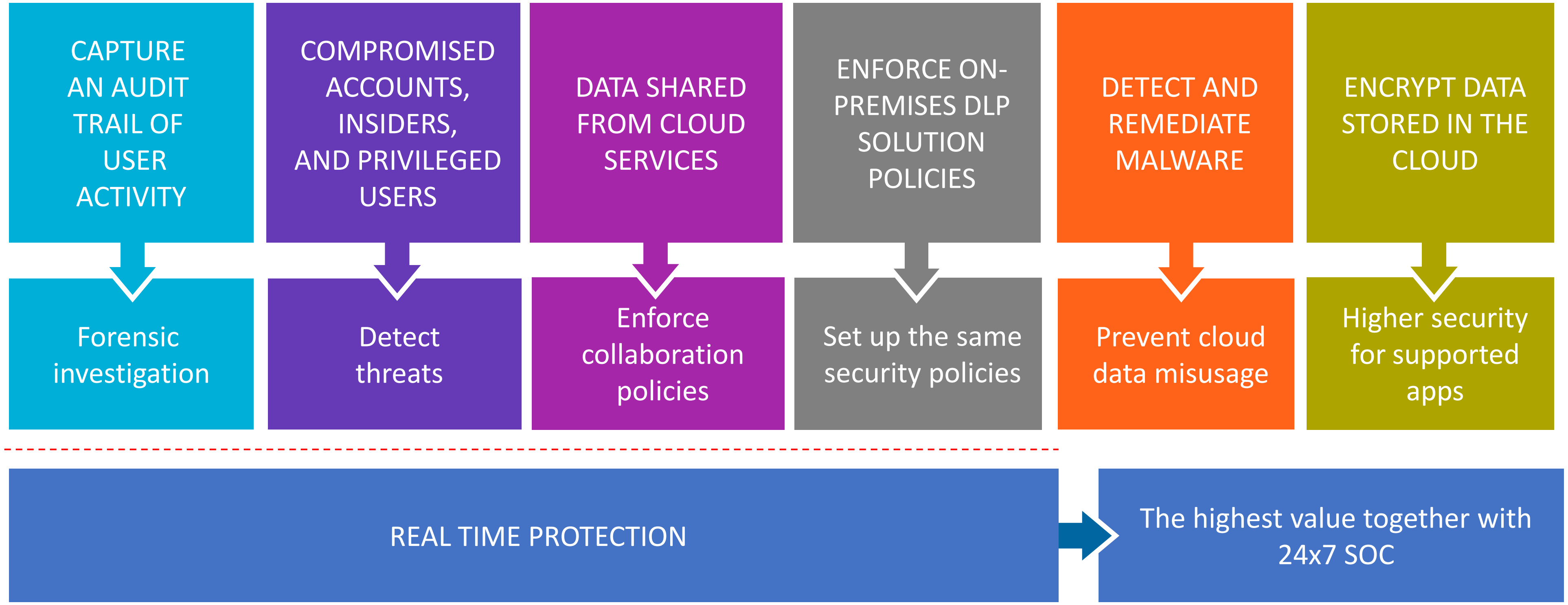
DISCOVER CLOUD SERVICES IN USE to have an overview if any company´s department is already using some cloud service but the CISO is not informed. Example: online convert of .doc to .pdf, not well known online storage, etc. Currently there is 25000 of known cloud services
ASSESS CLOUD SERVICE RISK shows you the overall information about the concrete cloud service credibility. with setting up of the company priorities and scales for the measures
DETECT DATA EXFILTRATION AND PROXY LEAKAGE: leverages the cloud as a vector for data exfiltration
Optional module: Reaction on the Shadow IT cloud services APPLYING CLOUD GOVERNANCE POLICIES.
Leverage the cloud services and allow the non-risky clouds

G.1d8 App Environment: Access

Cloud Access Security Broker



CASB: Sanctioned cloud use cases



CAPTURE AN AUDIT TRAIL OF USER ACTIVITY → Forensic investigation

COMPROMISED ACCOUNTS, INSIDERS, AND PRIVILEGED USERS → Detect threats

DATA SHARED FROM CLOUD SERVICES → Enforce collaboration policies

ENFORCE ON-PREMISES DLP SOLUTION POLICIES → Set up the same security policies

DETECT AND REMEDIATE MALWARE → Prevent cloud data misusage

ENCRYPT DATA STORED IN THE CLOUD → Higher security for supported apps

REAL TIME PROTECTION → The highest value together with 24x7 SOC

Sanctioned: If you approve the application for your organization's use because it meets all your security requirements.

Unsanctioned: If you don't recommend the application for your organization’s use because of its attack vulnerability.

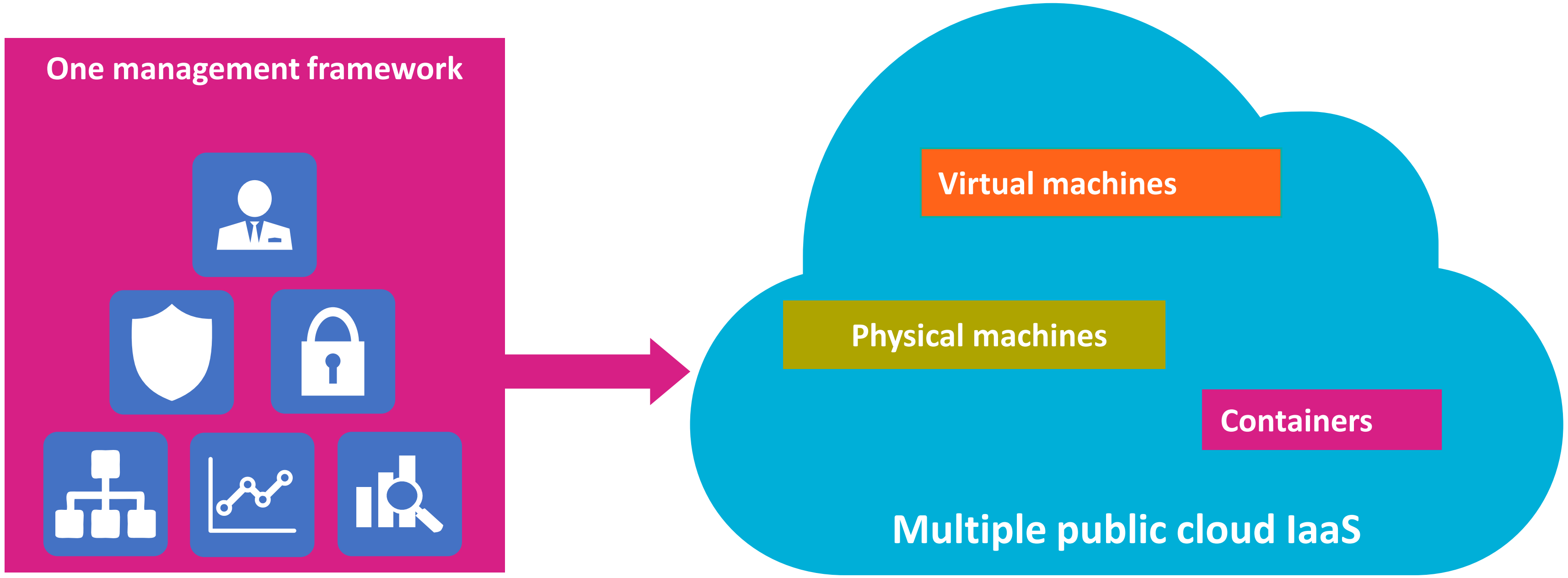
G.1d9 App Environment: Protection

Cloud Workload Protection Platform

CWPP: Sanctioned cloud use cases

to cover workloads across physical and virtual machines, containers and multiple public cloud IaaS, all from a single management framework and console.

CWP service centralize and automate reliable, proactive, and repeatable security controls without adding new labor costs.



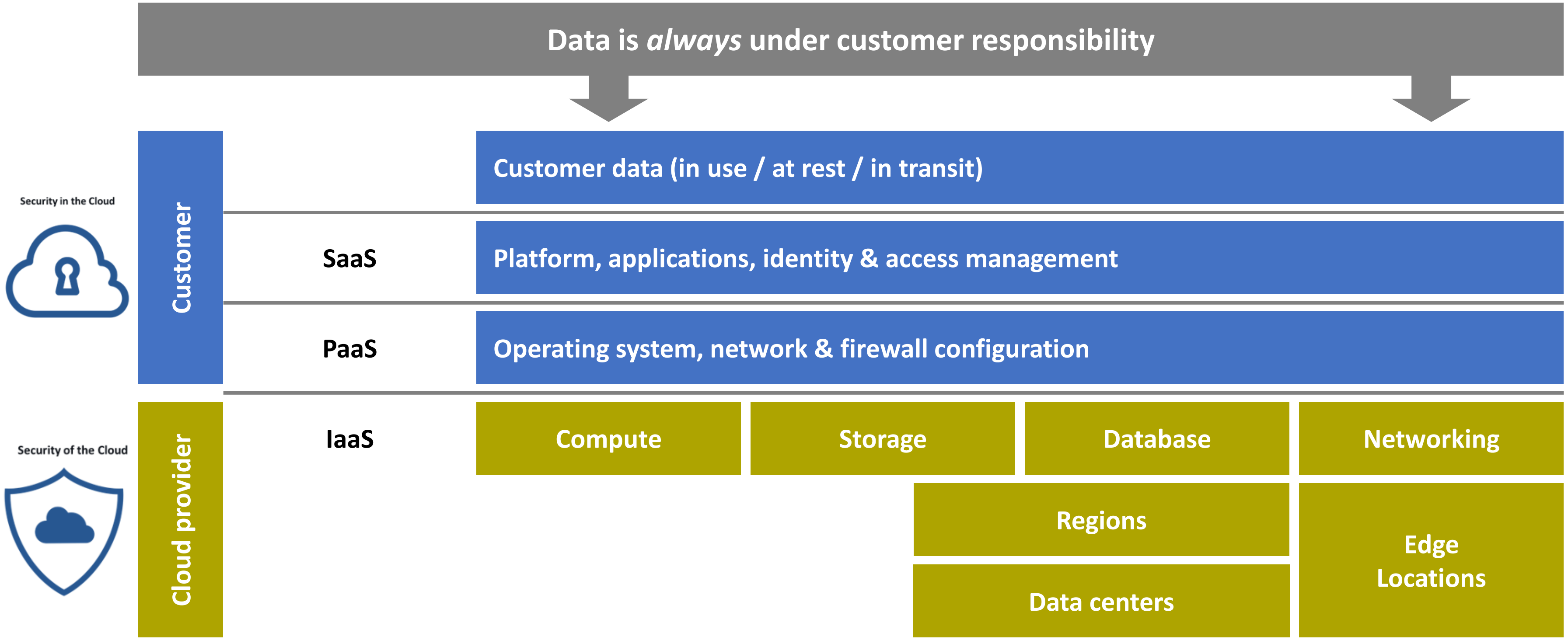
“The market for CWPPs is defined by workload-centric security protection solutions, which are typically agent-based (but not necessary). They address the unique requirements of server workload protection in modern hybrid data center architectures that span on-premises, physical and virtual machines (VMs) and multiple public cloud infrastructure as a service (IaaS) environments. Ideally, they also support container-based application architectures.” Gartner

G.1d10 App Environment: Protection

Cloud Workload Protection Platform

CWPP: Trust Relationship

Cloud security is a shared responsibility between the Infrastructure as a Service Cloud Providers like Azure, AWS, Google Cloud who focus on the security for their infrastructure with for example access to the infrastructure with authentication, DDOS protection, penetration testing, and secure storage.



Data is *always* under customer responsibility

Security in the Cloud

Customer

Customer data (in use / at rest / in transit)

SaaS — Platform, applications, identity & access management

PaaS — Operating system, network & firewall configuration

Security of the Cloud

Cloud provider

IaaS — Compute | Storage | Database | Networking

Regions

Data centers

Edge Locations

The customer is responsible for security in the cloud on the host OS, apps, local data stores.

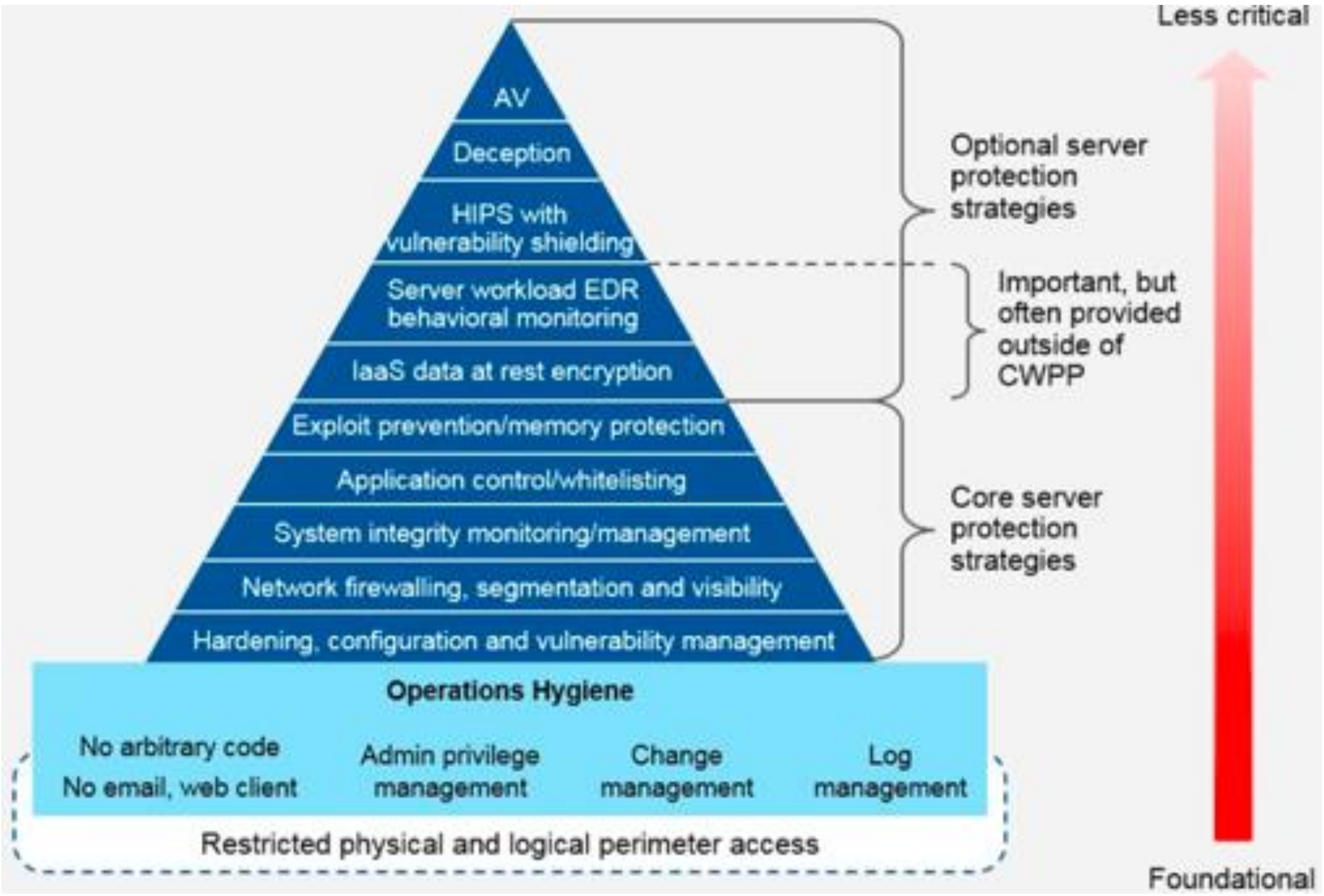
These Infrastructure as a Service Cloud Providers even state clearly on their web sites in some form or another like this diagram that the customer is responsible for the host and they are only responsible for the security of the cloud infrastructure.

G.1d11 App Environment: Protection

Cloud Workload Protection Platform

CWPP: Security Control Priorities

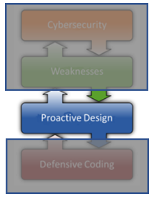
Recommended prioritization of security controls for hybrid cloud server workload protection



Less critical
AV
Deception
HIPS with vulnerability shielding
Server workload EDR behavioral monitoring
IaaS data at rest encryption
Exploit prevention/memory protection
Application control/whitelisting
System integrity monitoring/management
Network firewalling, segmentation and visibility
Hardening, configuration and vulnerability management
Operations Hygiene
No arbitrary code
No email, web client
Admin privilege management
Change management
Log management
Restricted physical and logical perimeter access
Optional server protection strategies
Important, but often provided outside of CWPP
Core server protection strategies
Foundational

G.1d12 App Environment: Protection

Cloud Workload Protection Platform

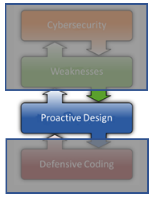


CWPP: Capabilities

Layer	Capability (Analysis - Description)	Trends (directions)												
		Cmpl Reqs	Pol Mgmt Consistency	Auto Pol Def & Provision	Usage-based Licensing	Segment Traffic East-West	Meltdown, Spectre	Get Sensitive Data Transaction	Container Scanning	Single Proc/Appl	Don't Rely on Signature	Serverless Prot	Encryption of Data at Rest	Supplemental Behavioural Monitoring
Mandatory	Hardening, configuration and vulnerability scanning	X	X											
	Workload segmentation, traffic visibility and optional network traffic encryption		X			X		X						
	System integrity monitoring/management		X	X					X					
	Application control		X	X						X	X	X		
	Exploit prevention and memory protection		X	X	X									
Important	IaaS data-at-rest encryption							X					X	
	Server EDR for behavioral monitoring													X
Optional	Host IPS including vulnerability-facing HIPS	X	X	X										
	Deception		X	X	X									
	Signature-based antivirus		X	X	X									

G.1e App Environment: Data

Data Protection

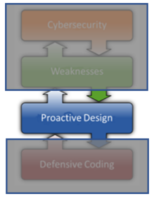


ZTA Pillar(s)	CSMA Tool	Name	Enforce	Enabling
Data	CKMS	Cloud Key Mgmt Service	Secure Key Mgmt	Centralized key control in hybrid cloud
Data	DLP	Data Loss Prevention	Detecting/Blocking Exfiltration	Protection of Company Data
Data	EDRM	Enterprise Digital Right Management	Blocking Data Usage	Protecting Intellectual Property

ZTDA: tools for implementing protection of data, making use of **cryptography**.

G.1e1 App Environment: Encryption

Cloud Key Management Service



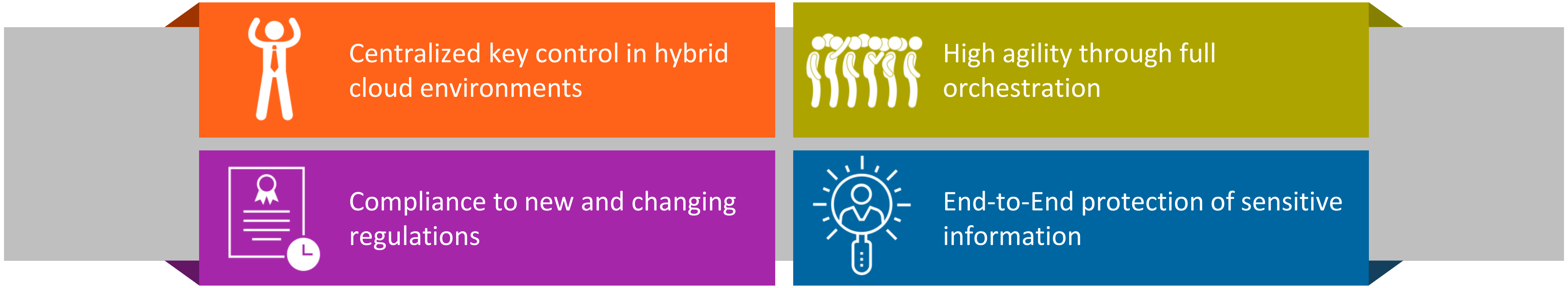
CKMS: Key Drivers

- Increasing **cloud adoption** by organizations – cost & flexibility
- Data is moving to **multi tenant environments** at third parties
- Increasing demand for **security level** equal to on premise protection
- Need for flexibility – workloads are shifted from private to public cloud or between public clouds
- Security needs to follow **workload agility**
- Transfer of **risk from** the **content to** the **keys** - key management is a vital function in cloud encryption
- Constant change of the **legal** and **regulatory environment**

G.1e2 App Environment: Encryption

Cloud Key Management Service

CKMS: Secure and Control Access to Data in Cloud



Centralized key control in hybrid cloud environments

High agility through full orchestration

Compliance to new and changing regulations

End-to-End protection of sensitive information

Clients increasingly find themselves needing to secure cloud deployments in multiple public clouds – Gartner, 2018Q2

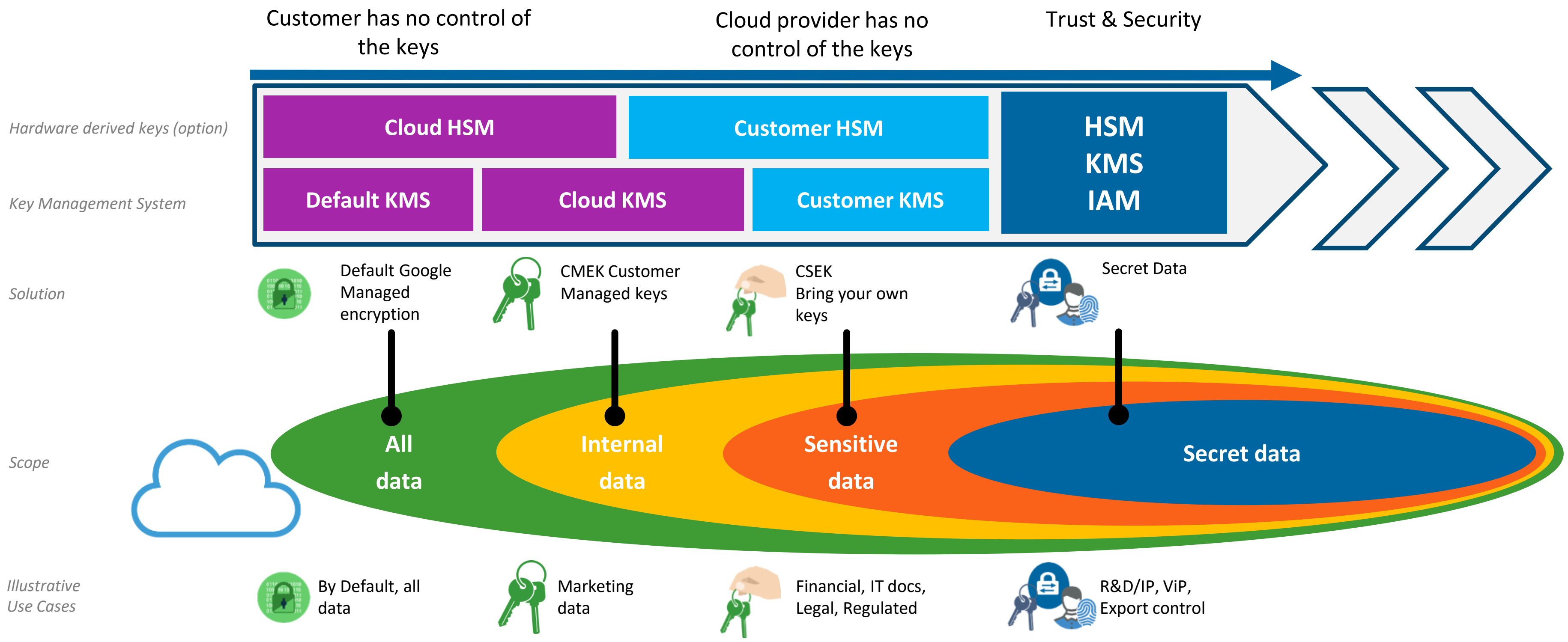
Encryption of data at rest should be considered a mandatory best practice for public-cloud-based servers - Gartner - Market Guide for Cloud Workload Protection Platforms, 2018Q1

As the use of encryption grows, centralized cloud key management will become increasingly desirable, and even necessary – Gartner - Prioritize Enterprise Wide Encryption for Critical Datasets, 2017Q2

G.1e2 App Environment: Encryption

Cloud Key Management Service

CKMS: Trust vs Agility



Customer has no control of the keys

Cloud provider has no control of the keys

Trust & Security

Hardware derived keys (option): Cloud HSM | Customer HSM

Key Management System: Default KMS | Cloud KMS | Customer KMS

HSM
KMS
IAM

Solution:
- Default Google Managed encryption
- CMEK Customer Managed keys
- CSEK Bring your own keys
- Secret Data

Scope: All data | Internal data | Sensitive data | Secret data

Illustrative Use Cases:
- By Default, all data
- Marketing data
- Financial, IT docs, Legal, Regulated
- R&D/IP, ViP, Export control

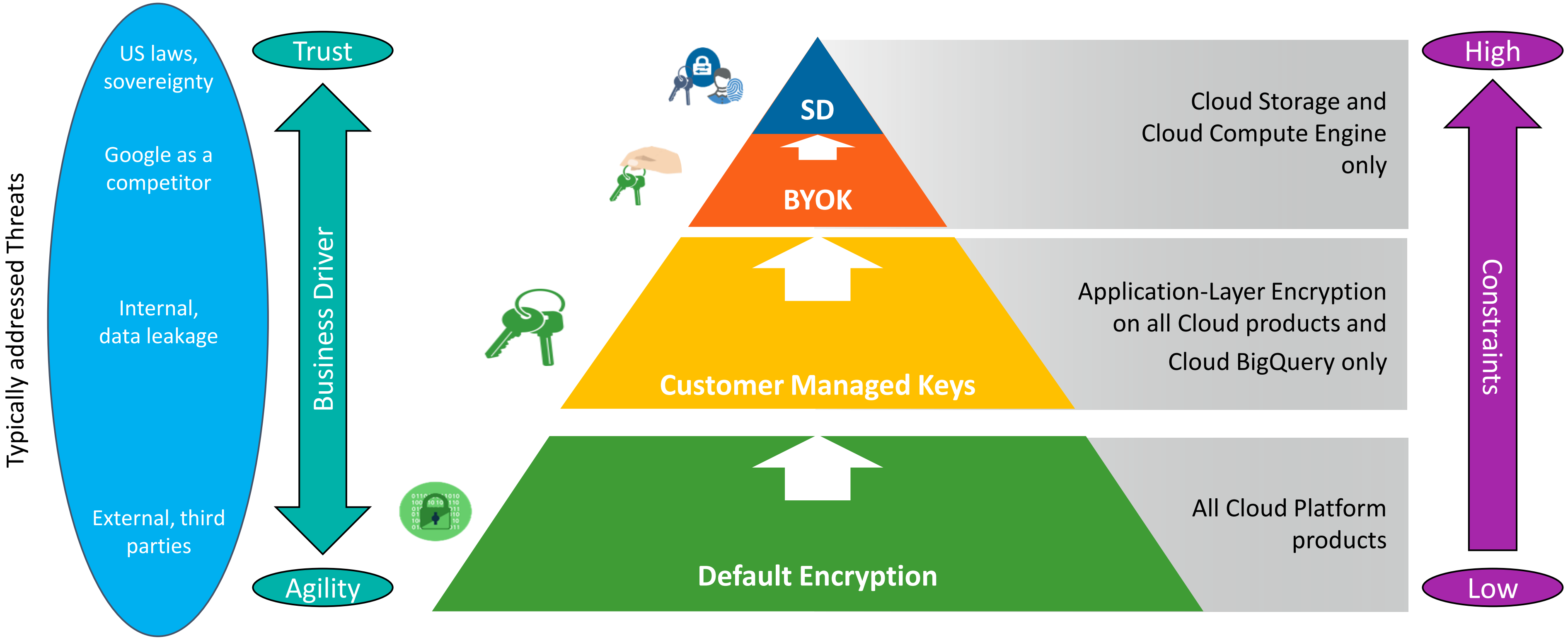
Agility: all the key managed by the Cloud provider

Trust: the key managed by the customer

G.1e3 App Environment: Encryption

Cloud Key Management Service

CKMS: Trust vs Agility



Typically addressed Threats

US laws, sovereignty

Google as a competitor

Internal, data leakage

External, third parties

Trust

Business Driver

Agility

SD

BYOK

Customer Managed Keys

Default Encryption

Cloud Storage and Cloud Compute Engine only

Application-Layer Encryption on all Cloud products and Cloud BigQuery only

All Cloud Platform products

High

Constraints

Low

Trust: the key managed by the customer

Agility: all the key managed by the Cloud provider

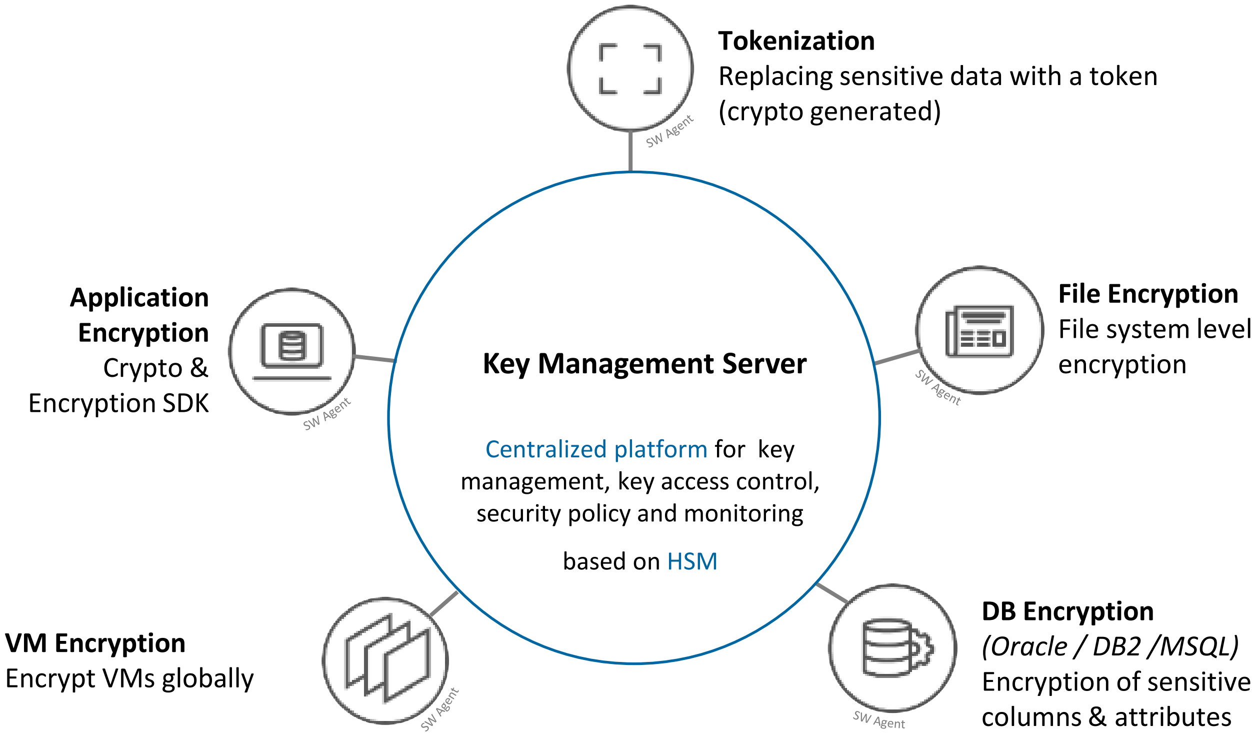
G.1e4 App Environment: Encryption

Cloud Key Management Service

CKMS: Crypto Foundation

- **Interface**: same integration among components (O.S., Service, Applications etc.)

- **Protocol**: Homogeneity of cipher initiatives



Key Management Server

Centralized platform for key management, key access control, security policy and monitoring

based on HSM

Tokenization
Replacing sensitive data with a token (crypto generated)

File Encryption
File system level encryption

DB Encryption
(Oracle / DB2 /MSQL)
Encryption of sensitive columns & attributes

VM Encryption
Encrypt VMs globally

Application Encryption
Crypto & Encryption SDK

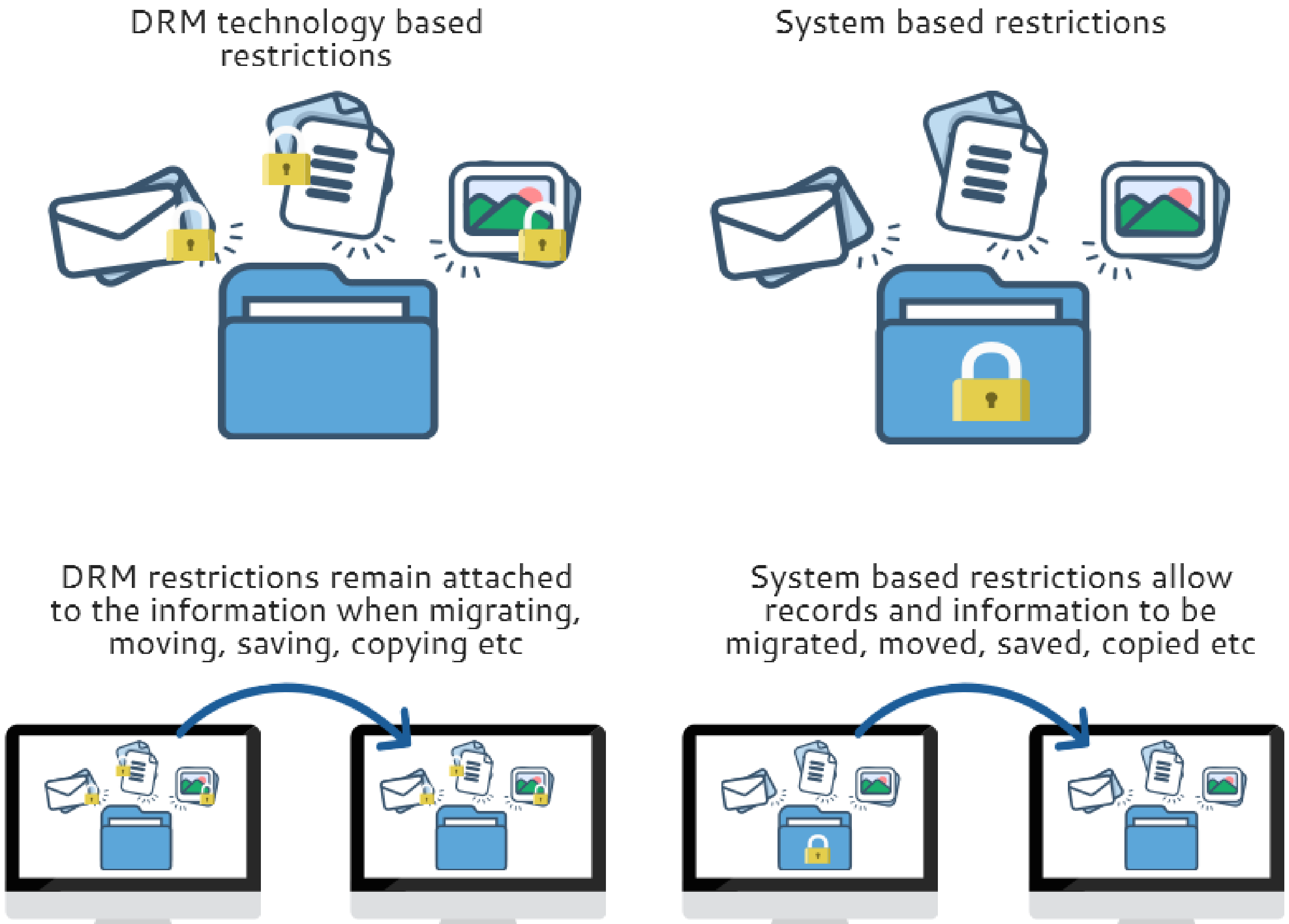
Data Masking: Static (**SDM**) and Dynamic (**DDM**): it could be performed by DB enabled features (e.g. MS SQL, Oracle etc.) or external products. (see Gartner)

"Data masking can dynamically or statically protect sensitive data by replacing it with fictitious data that looks realistic to prevent data loss in different use cases. This research will aid CISOs in selecting the appropriate technologies for their needs." – Gartner - Static and Dynamic Data Masking Explained, 20 October 2015 (https://www.gartner.com/en/documents/3153926)

G.1e5 App Environment: Right Management

Enterprise Digital Right Management

EDRM: Restriction to Information usage



DRM technology based restrictions

System based restrictions

DRM restrictions remain attached to the information when migrating, moving, saving, copying etc

System based restrictions allow records and information to be migrated, moved, saved, copied etc

way to protect copyrights for digital media. This approach includes the use of technologies that limit the copying and use of copyrighted works and proprietary software.

- **Consumer version**: Digital Rights Management (**DRM**): Music, movies, e-books
- **Enterprise Version**: Enterprise Right Management (**ERM**): Intellectual Property

See Queensland Government: https://www.forgov.qld.gov.au/information-and-communication-technology/recordkeeping-and-information-management/recordkeeping/store-protect-and-care-for-records/store-protect-and-care-for-digital-records/use-digital-rights-management-and-encryption

G.1e6 App Environment: Right Management

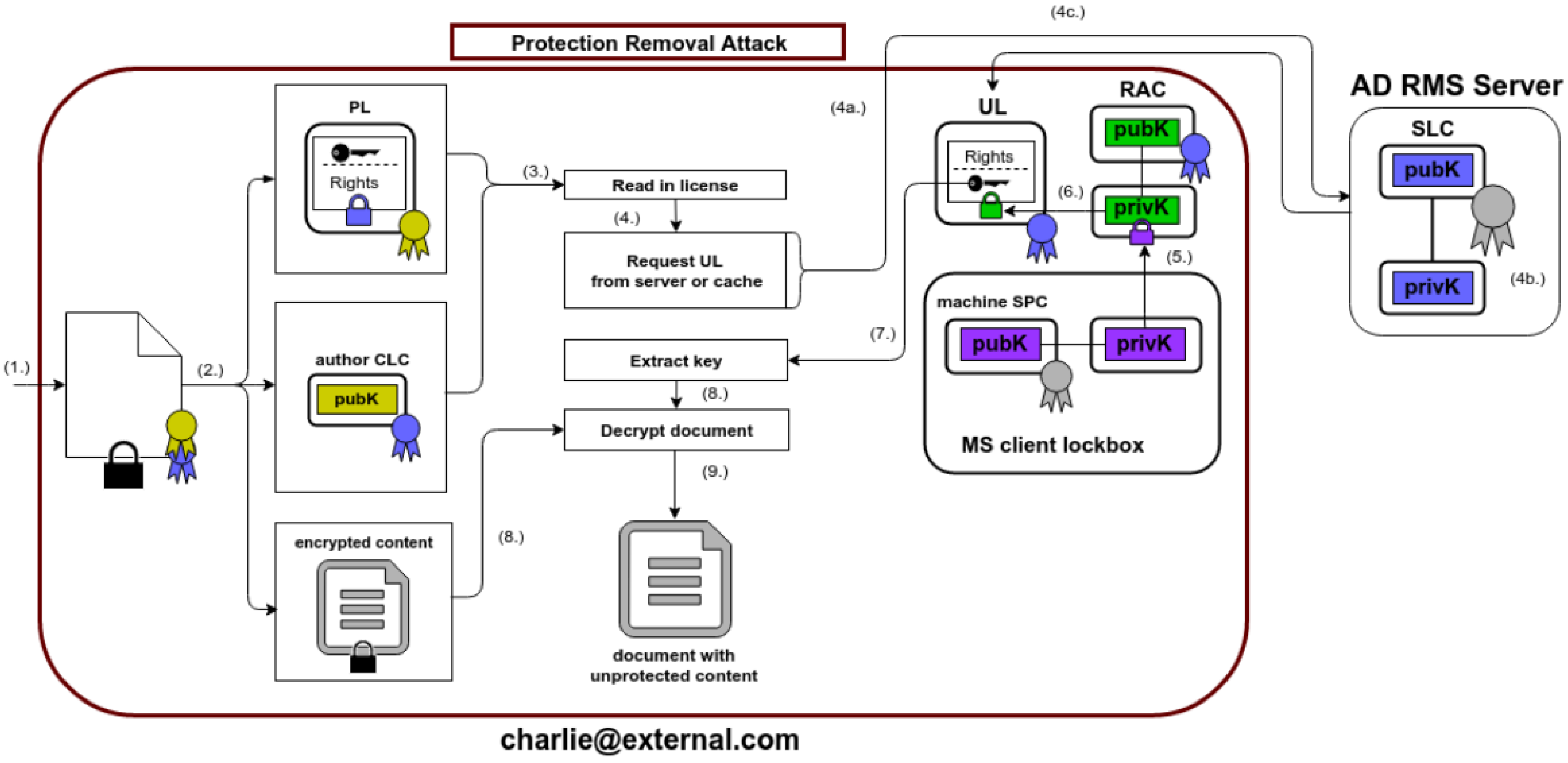
Enterprise Digital Right Management

DisARMS: Disabeling Attacks on Rights Management Services

Security analysis of Microsoft RMS and present two working attacks:

1. **completely removed the RMS protection** of a Word document previously having a **view-only permission ➔** MS **RMS** can only **enforce all-or-nothing access**.
2. **attack extended** to be **stealthy** in the following sense: We show how to modify the content of an RMS write-protected Word document claiming to be write protected.

Responsibly disclosed: MSRC Case 33210



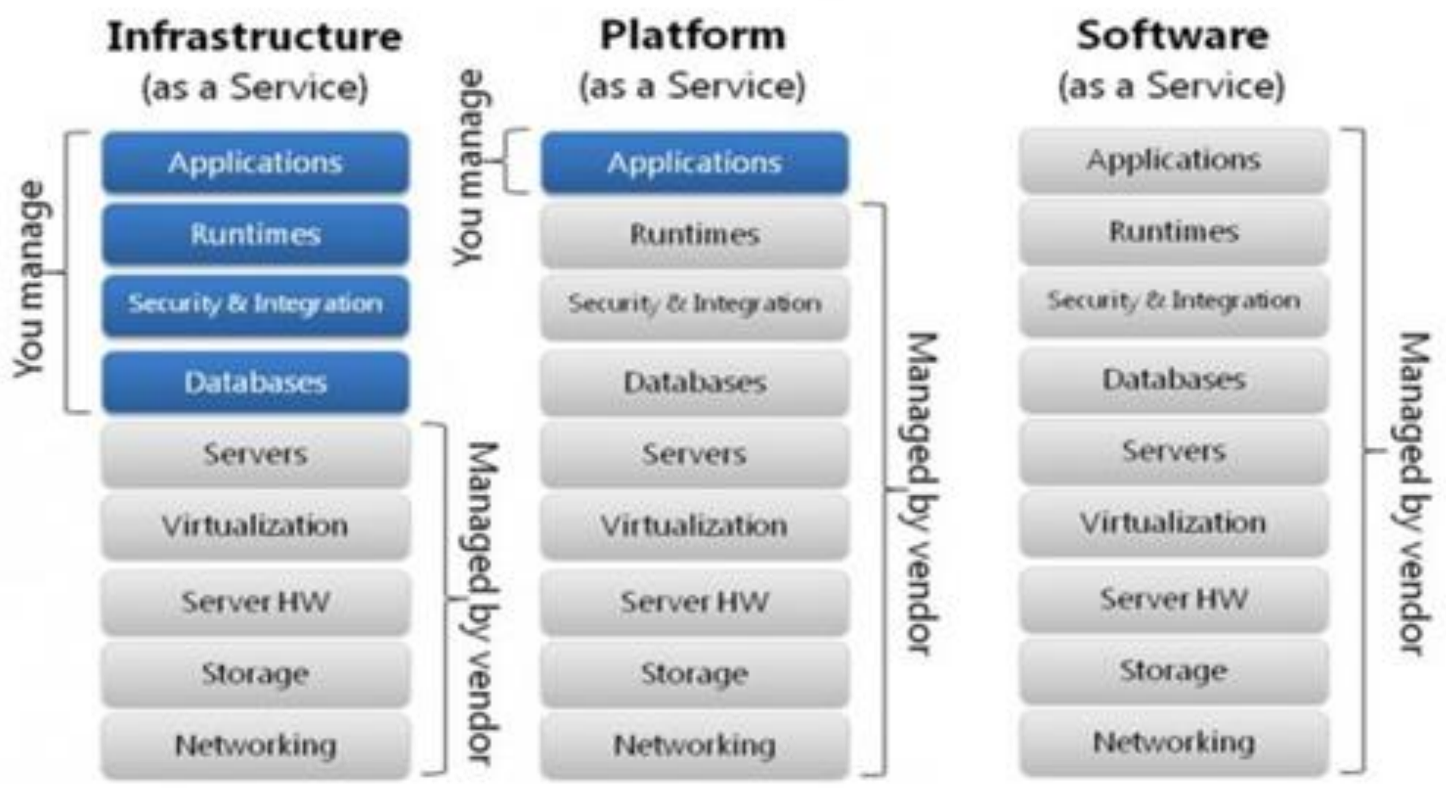
For more information: https://www.usenix.org/system/files/conference/woot16/woot16-paper-grothe.pdf
https://www.usenix.org/sites/default/files/conference/protected-files/woot16_slides_grothe.pdf

G.2 Cloud: Service Provider

Shared Responbibility Model

The NIST published in 2011 the SP800-145, providing thedefinition of the 3 major delivery models

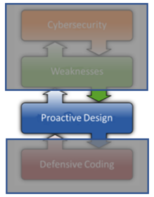
- **IaaS**: base infrastructure (Virtual machine, Software Define Network, Storage attached). End user have to configure and manage platform and environment, deploy applications on it
- **PaaS**: platform allowing end user to develop, run, and manage applications without the complexity of building and maintaining the infrastructure.
- **SaaS**: “on-demand software”. Typically accessed by users using a thin client via a web browser. In SaaS everything can be managed by vendors: applications, runtime, data, middleware, OSes, virtualization, servers, storage and networking, End users have to use it



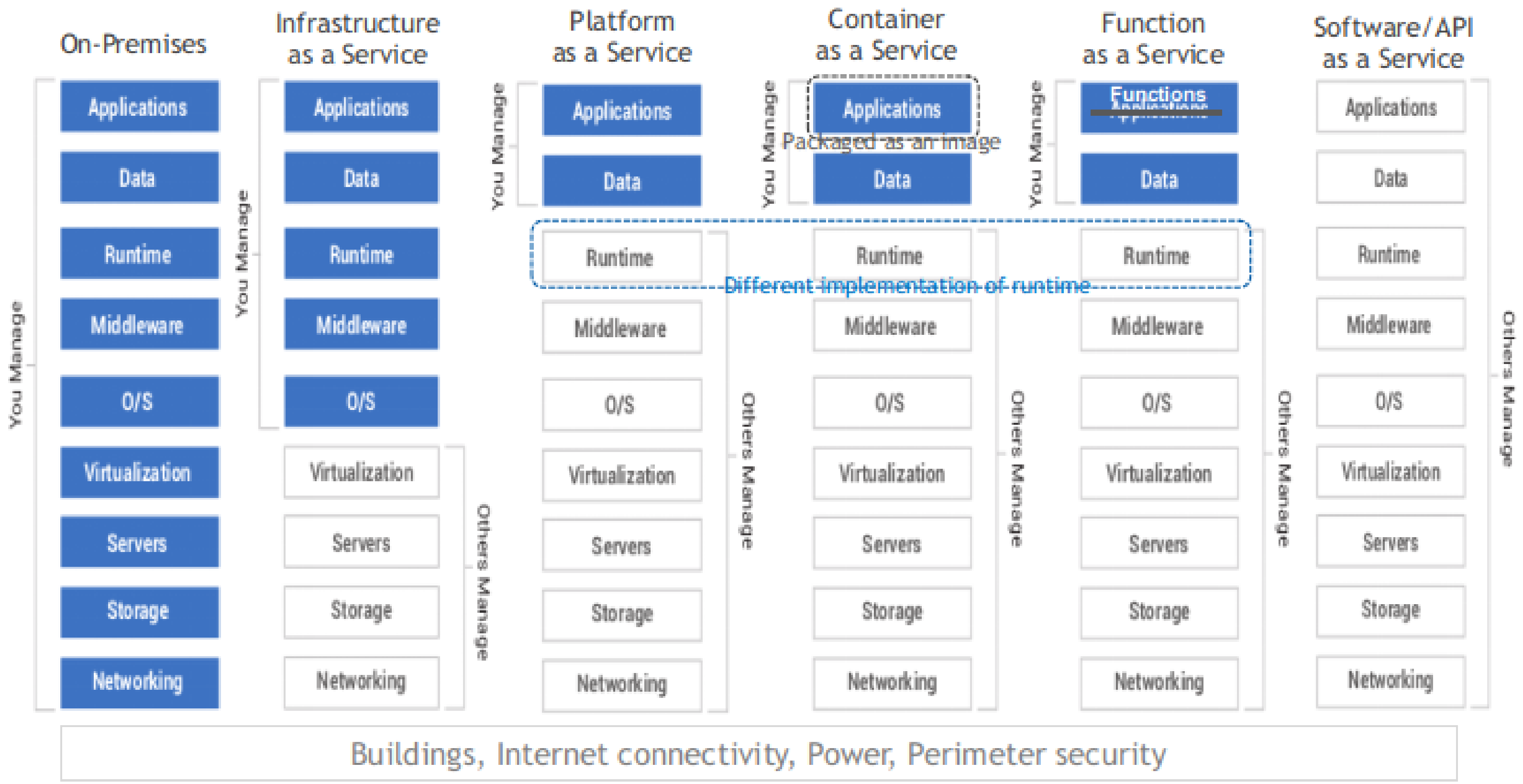
Each service comes with different responsibilities shared among Cloud Vendor and Customer

G.2a Cloud: Service Provider

Shared Responbibility Model: refinements



The initial shared responsibility model was further refined, splitting the PaaS model in 3 more detailed services, introducing as new interfaces the containers (CaaS) and the API (FaaS)

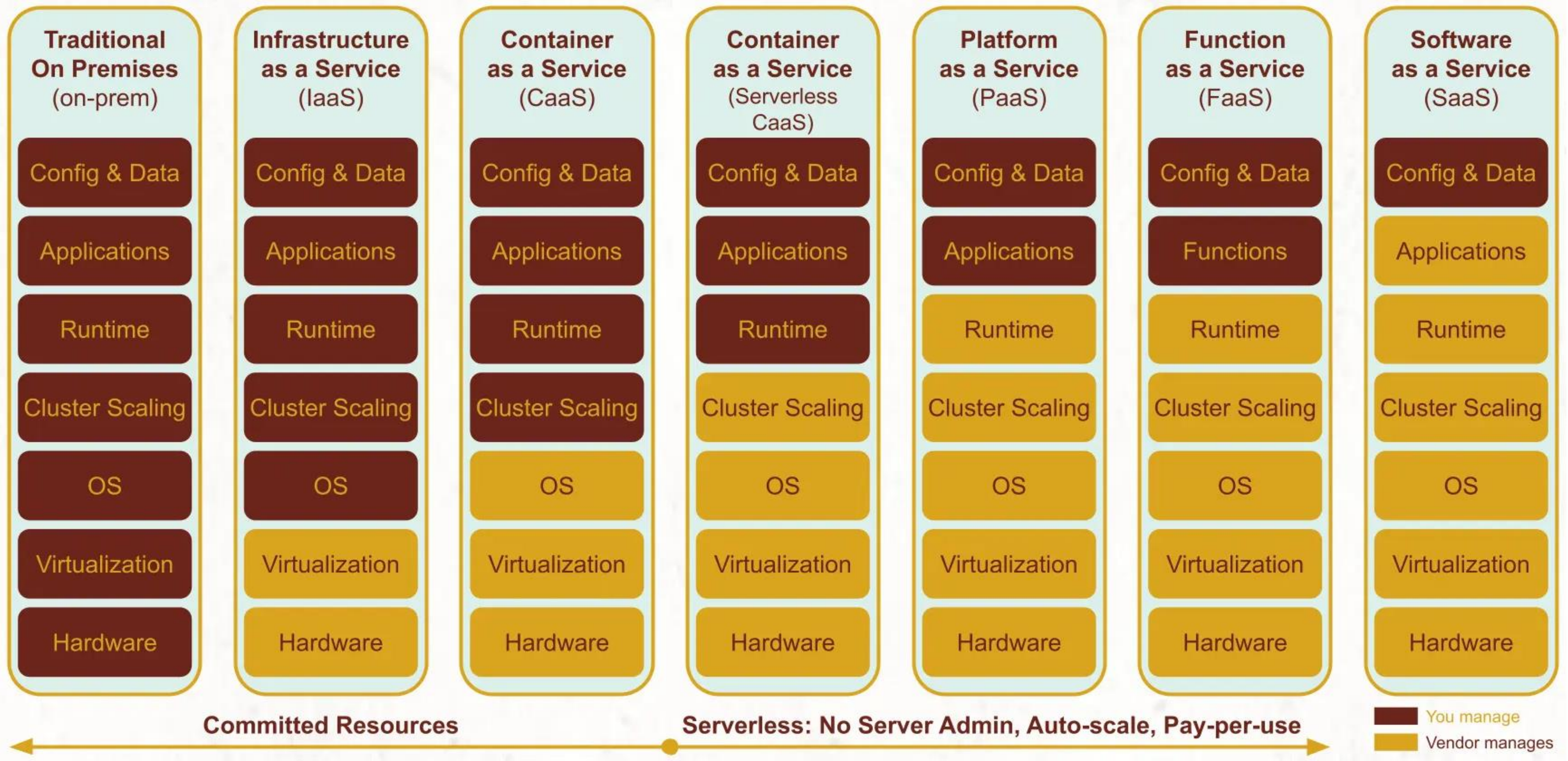


- **PaaS**: has morphed into two more services, container (CaaS) and function (FaaS).
- **CaaS**: bundle of application, usually shipped as Docker images, organized in cluster (usually managed by Kubernetes)
- **FaaS**: functions “on-demand software”. It allows developers to write code in small units and the service will manage everything else

G.2b Cloud: Service Provider

Play per Use

Serverless could be see as having serve managed by the provider:



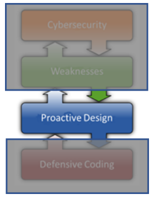
Traditional On Premises (on-prem)	Infrastructure as a Service (IaaS)	Container as a Service (CaaS)	Container as a Service (Serverless CaaS)	Platform as a Service (PaaS)	Function as a Service (FaaS)	Software as a Service (SaaS)
🟥 Config & Data	🟥 Config & Data	🟥 Config & Data	🟥 Config & Data	🟥 Config & Data	🟥 Config & Data	🟥 Config & Data
🟥 Applications	🟥 Applications	🟥 Applications	🟥 Applications	🟥 Applications	🟥 Functions	🟨 Applications
🟥 Runtime	🟥 Runtime	🟥 Runtime	🟥 Runtime	🟨 Runtime	🟨 Runtime	🟨 Runtime
🟥 Cluster Scaling	🟥 Cluster Scaling	🟥 Cluster Scaling	🟨 Cluster Scaling	🟨 Cluster Scaling	🟨 Cluster Scaling	🟨 Cluster Scaling
🟥 OS	🟥 OS	🟨 OS	🟨 OS	🟨 OS	🟨 OS	🟨 OS
🟥 Virtualization	🟨 Virtualization	🟨 Virtualization	🟨 Virtualization	🟨 Virtualization	🟨 Virtualization	🟨 Virtualization
🟥 Hardware	🟨 Hardware	🟨 Hardware	🟨 Hardware	🟨 Hardware	🟨 Hardware	🟨 Hardware

Committed Resources ← → Serverless: No Server Admin, Auto-scale, Pay-per-use

🟥 You manage
🟨 Vendor manages

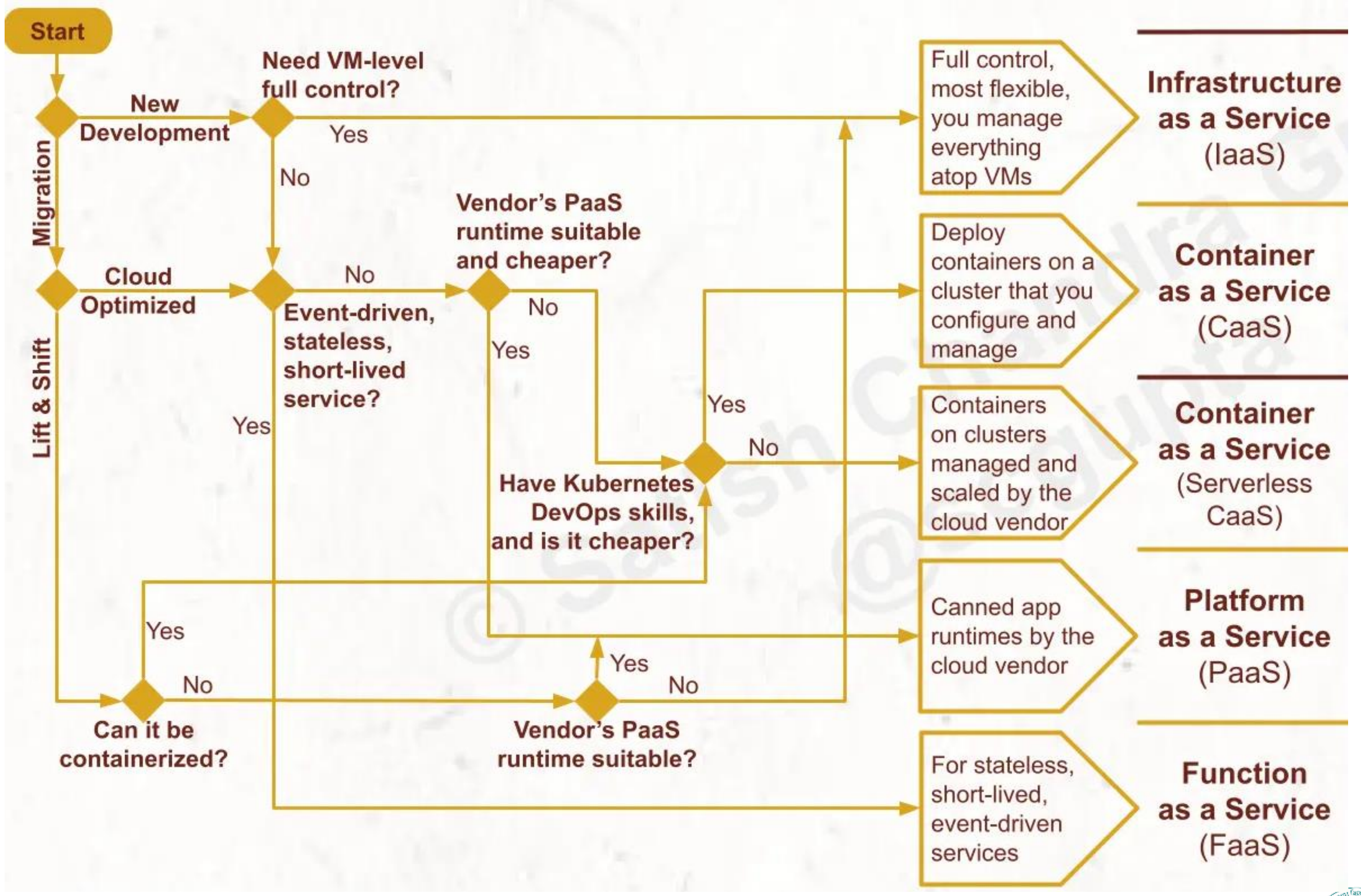
G.2c Cloud: Service Provider

Migration to cloud



Redesigning the app and optimizing it for the cloud, brings to the same situation as developing a new app with two caveats:

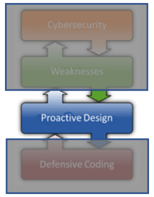
- **VM control level**: nothing to redesign. You will be going to use IaaS, so stick to lift-and-shift.
- **Lift and Shift**: If the application can be containerized, then you must define containers for it and enjoy the benefits of a uniform runtime. The choice between CaaS and Serverless CaaS is the same. Choose CaaS if you have Kubernetes skills in your DevOps team and CaaS is cheaper, else go for Serverless CaaS. If the app can't be containerized but one of the PaaS runtimes suits it, then go for PaaS, else you have to go for IaaS



Start
Migration
Lift & Shift
New Development
Need VM-level full control?
Yes
No
Cloud Optimized
No
Event-driven, stateless, short-lived service?
Yes
Vendor's PaaS runtime suitable and cheaper?
No
Yes
Yes
No
Have Kubernetes DevOps skills, and is it cheaper?
Yes
No
Can it be containerized?
Yes
No
Vendor's PaaS runtime suitable?
Full control, most flexible, you manage everything atop VMs
Infrastructure as a Service (IaaS)
Deploy containers on a cluster that you configure and manage
Container as a Service (CaaS)
Containers on clusters managed and scaled by the cloud vendor
Container as a Service (Serverless CaaS)
Canned app runtimes by the cloud vendor
Platform as a Service (PaaS)
For stateless, short-lived, event-driven services
Function as a Service (FaaS)

G.3 Containers: Images and Orchestration

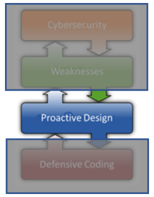
Automate Security into CI/CD Pipelines with Jenkins- Introduction to DevSecOps



1. **Containerization History**: Bringing Standardization as in Transportation Industry
2. **Docker Architecture**: How Docker works
3. **Docker Containers**: Management
4. **Docker Images**: Registry and Management
5. **Container Orchestration**: multiple containers in different environments

G.3 Containers: Images and Orchestration

Automate Security into CI/CD Pipelines with Jenkins- Introduction to DevSecOps



Analogy: Transportation and SW Development

G.3.1a Containers: History

Bringing Strandardization to Transportation Industry with Containers

1. **< 1960s**: Different Size, Shape , Consistence, Weights. Manual, specific work for (un)loading

2. **> 1960s**: Same Size, Shape, Consistence and Weight (of the Containers). Automatable work for (un)loading, stacking and transportation (without being opened!)

Benefits ➔ Reduction of

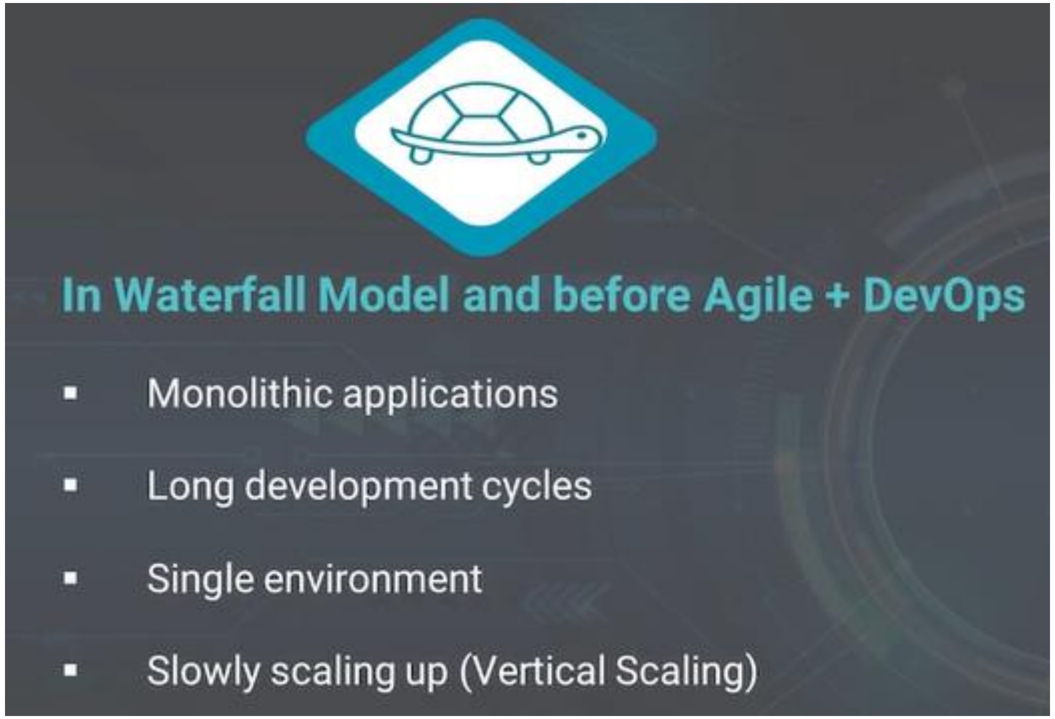
- **Time for (un)loading**
- **# Incidents**
- **Transportation Costs**

G.3.1b Containers: History

Containers in IT world

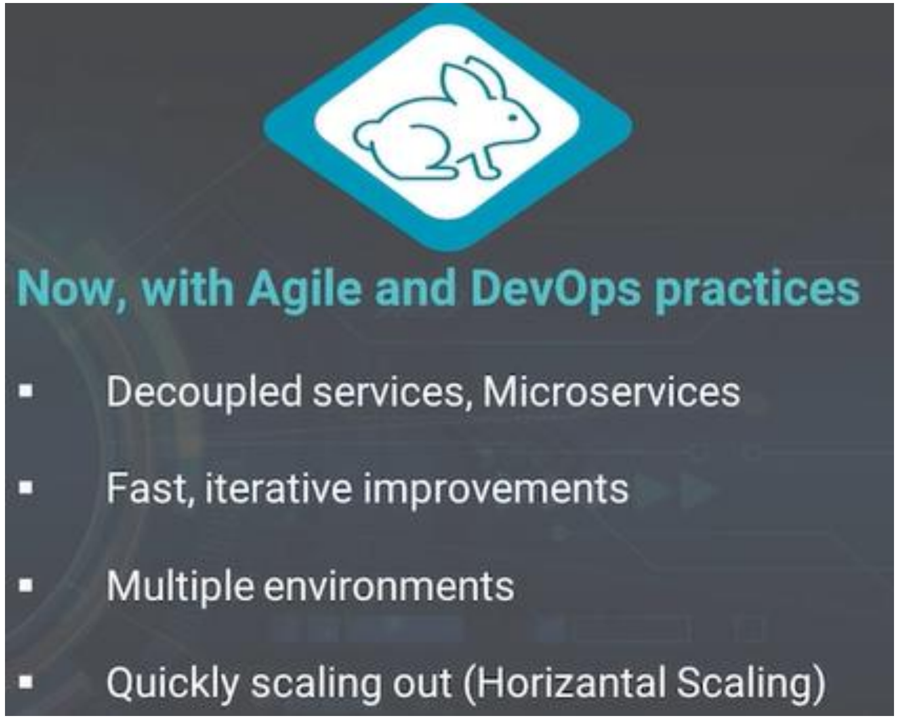
Delivery Speed Problem: old working way is not helping in coop with market speed

Time for (un)loading



In Waterfall Model and before Agile + DevOps

- Monolithic applications
- Long development cycles
- Single environment
- Slowly scaling up (Vertical Scaling)



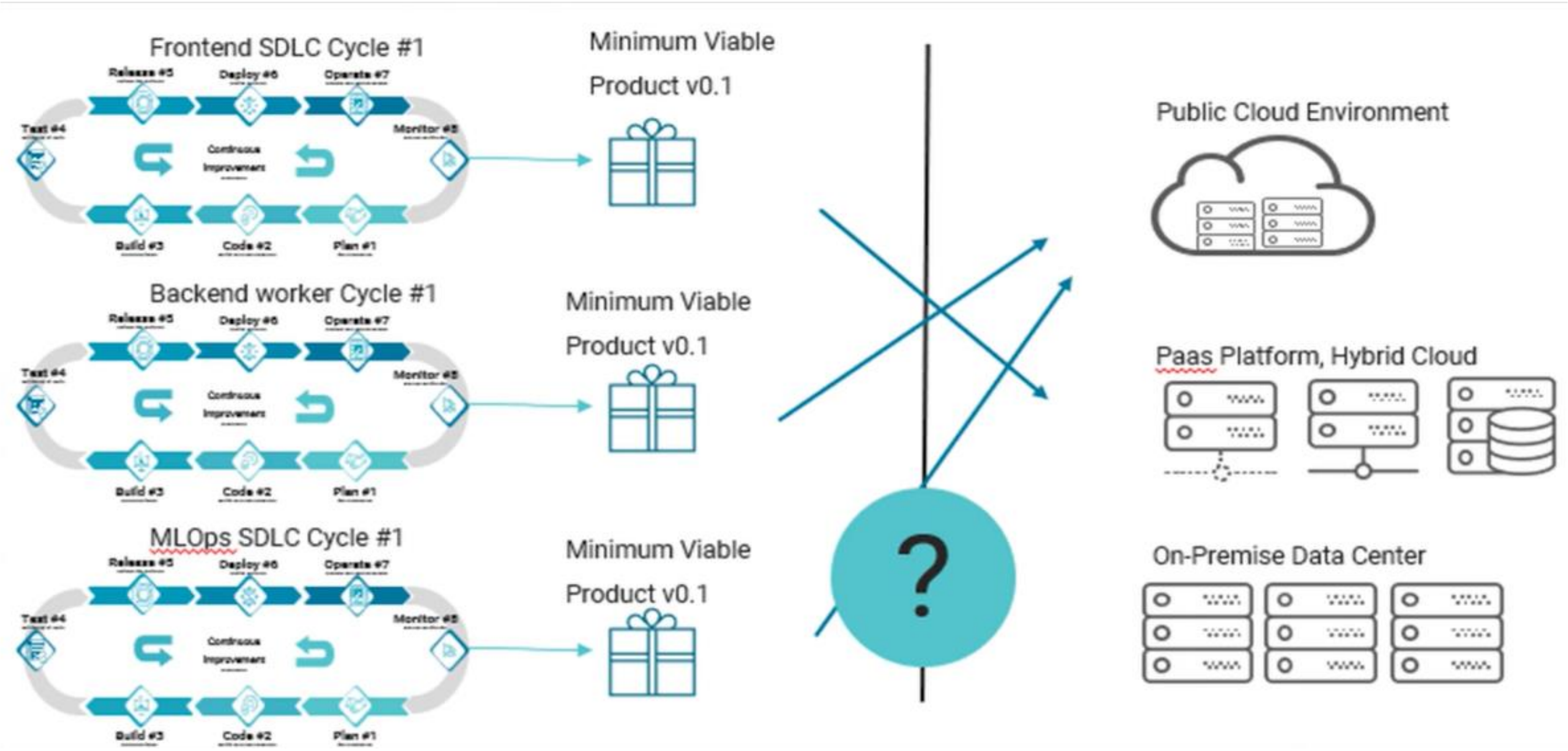
Now, with Agile and DevOps practices

- Decoupled services, Microservices
- Fast, iterative improvements
- Multiple environments
- Quickly scaling out (Horizantal Scaling)

G.3.1c Containers: History

Containers in IT world

Deployment Problem: Different Environments (like Architecture, CI/CD, Skills, Tiers, etc)

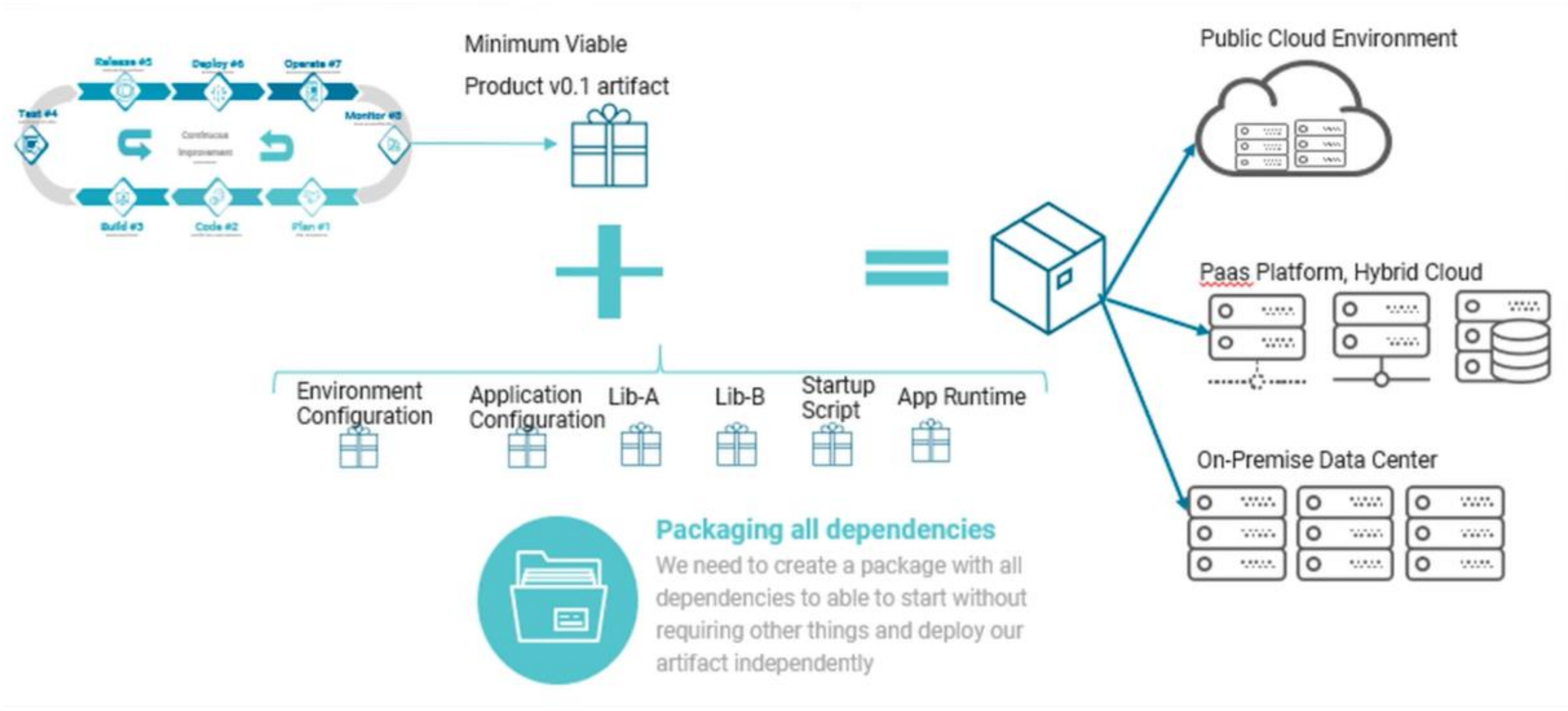


Transportation Costs

G.3.1d Containers: History

Containers in IT world

Matrix Dependency Problem: Libraries, Configuration Scripts,



Packaging all dependencies

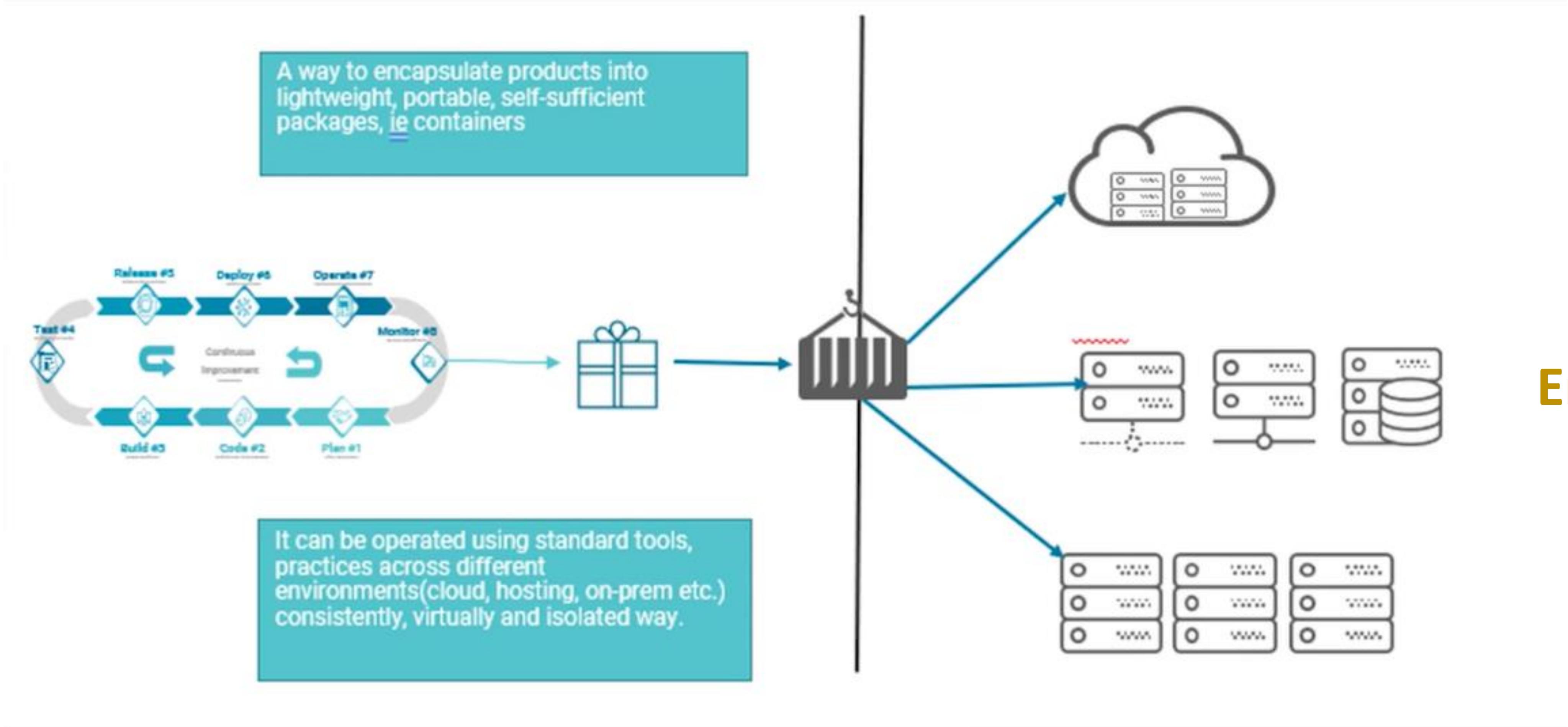
We need to create a package with all dependencies to able to start without requiring other things and deploy our artifact independently

# Incidents

G.3.1e Containers: History

Containers in IT world

Container System for Application: encapsulating the product itself, ready for different environment



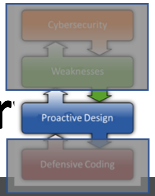
A way to encapsulate products into lightweight, portable, self-sufficient packages, ie containers

It can be operated using standard tools, practices across different environments(cloud, hosting, on-prem etc.) consistently, virtually and isolated way.

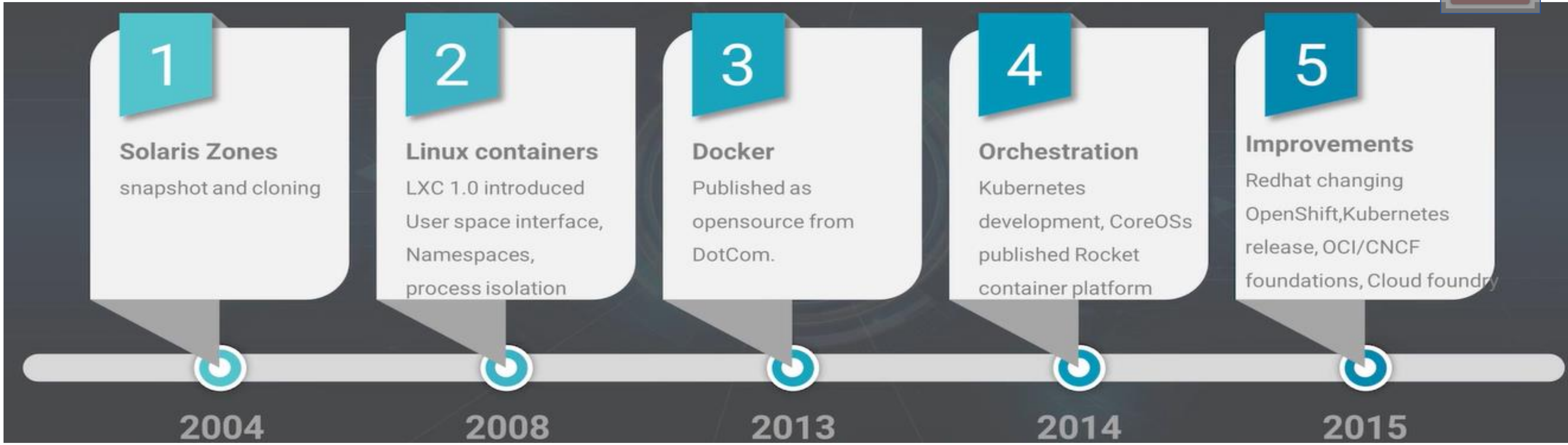
Isolating from Environment

G.3.2a Containers: Docker Architecture

evolution of Container Technology



Container Histor



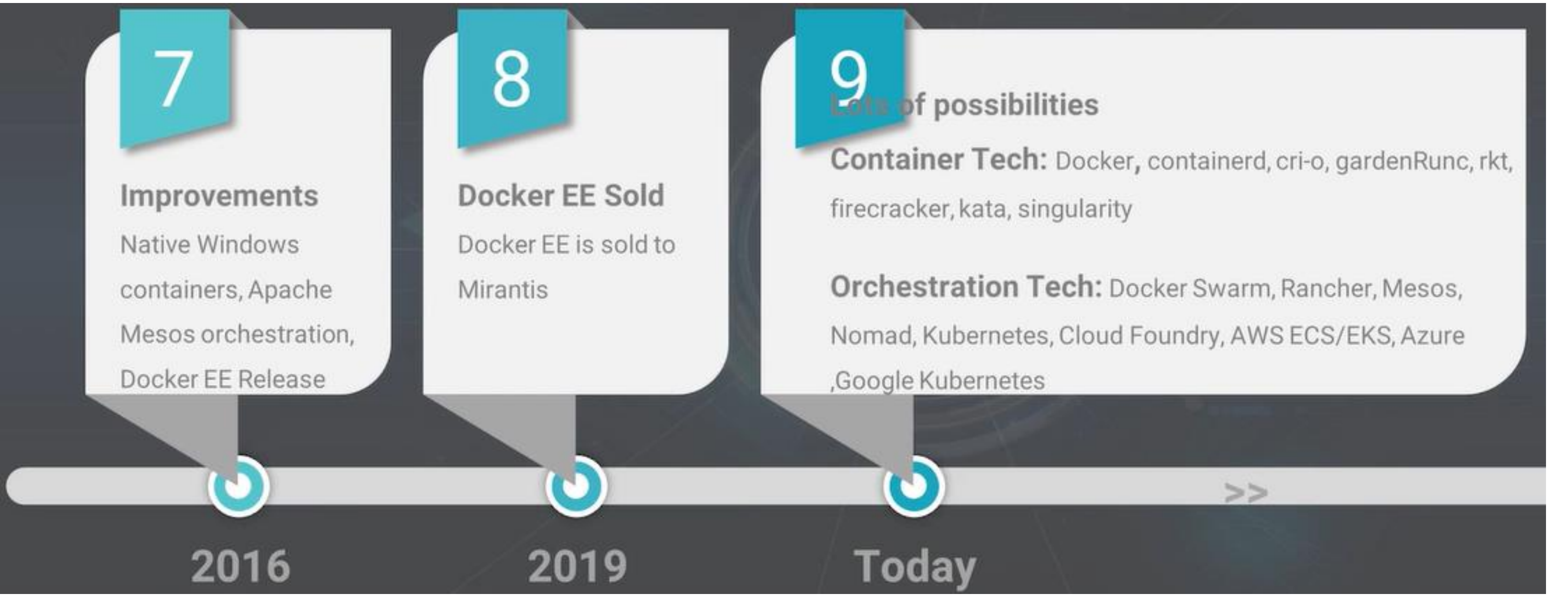
1 **Solaris Zones** snapshot and cloning — 2004

2 **Linux containers** LXC 1.0 introduced User space interface, Namespaces, process isolation — 2008

3 **Docker** Published as opensource from DotCom. — 2013

4 **Orchestration** Kubernetes development, CoreOSs published Rocket container platform — 2014

5 **Improvements** Redhat changing OpenShift,Kubernetes release, OCI/CNCF foundations, Cloud foundry — 2015



7 **Improvements** Native Windows containers, Apache Mesos orchestration, Docker EE Release — 2016

8 **Docker EE Sold** Docker EE is sold to Mirantis — 2019

9 **Lots of possibilities** — Today

Container Tech: Docker**,** containerd, cri-o, gardenRunc, rkt, firecracker, kata, singularity

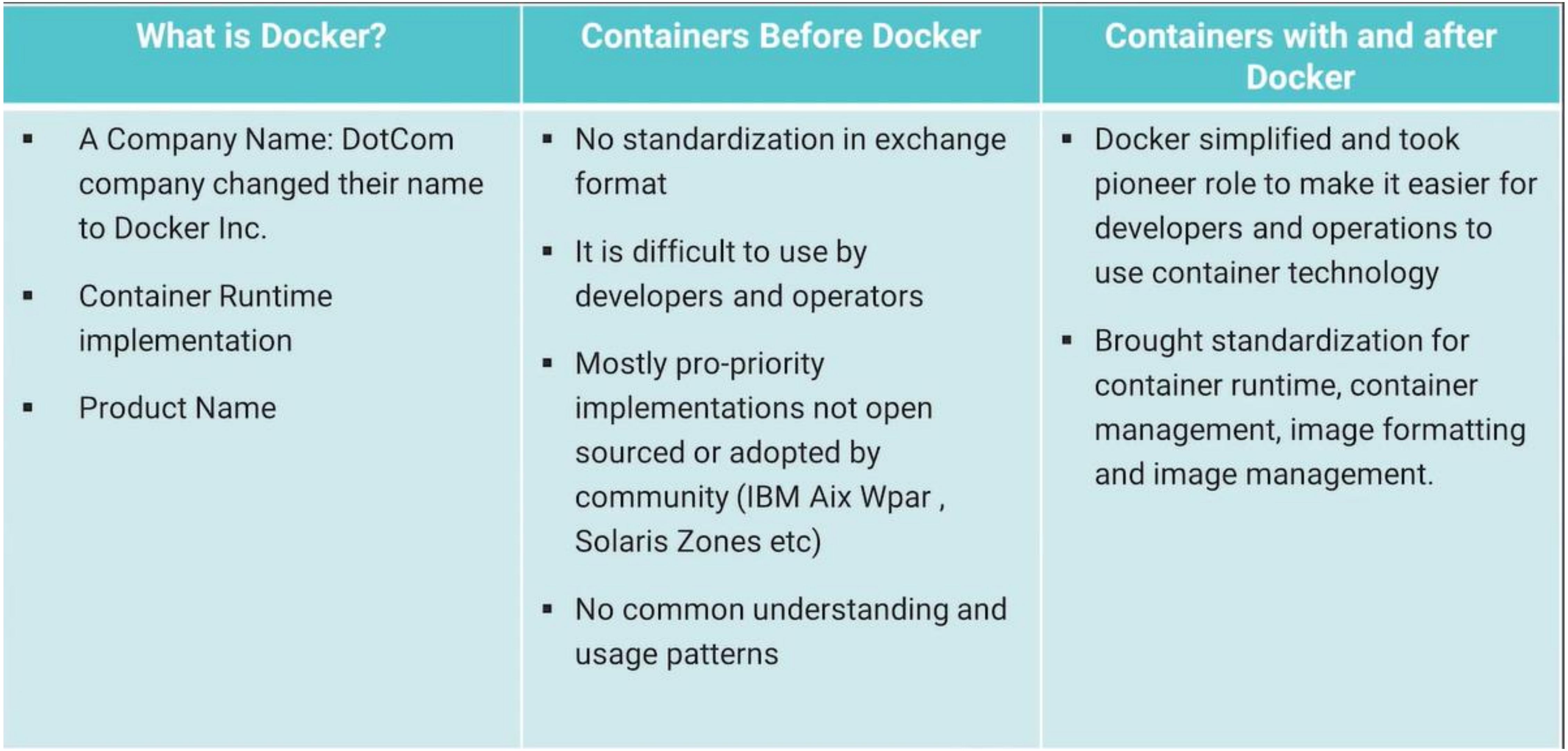
Orchestration Tech: Docker Swarm, Rancher, Mesos, Nomad, Kubernetes, Cloud Foundry, AWS ECS/EKS, Azure ,Google Kubernetes

CNCF landscape page: https://landscape.cncG.io

G.3.2b Containers: Docker Architecture

What does Docker Mean?

The historic importance of Docker: no more needs for admin guy, the developers could do all the stuff theirselves

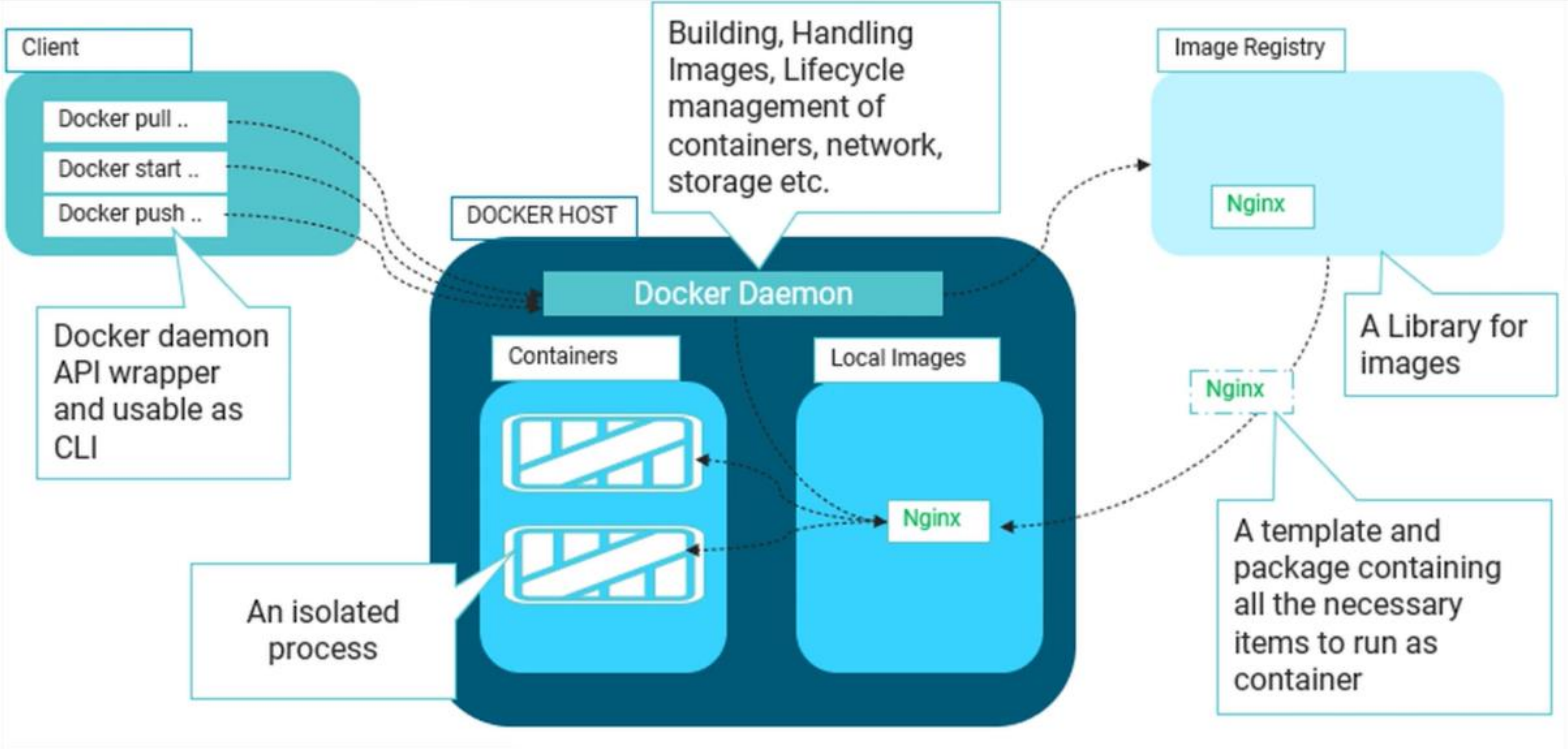


What is Docker?	Containers Before Docker	Containers with and after Docker
▪ A Company Name: DotCom company changed their name to Docker Inc. ▪ Container Runtime implementation ▪ Product Name	▪ No standardization in exchange format ▪ It is difficult to use by developers and operators ▪ Mostly pro-priority implementations not open sourced or adopted by community (IBM Aix Wpar , Solaris Zones etc) ▪ No common understanding and usage patterns	▪ Docker simplified and took pioneer role to make it easier for developers and operations to use container technology ▪ Brought standardization for container runtime, container management, image formatting and image management.

G.3.2c Containers: Docker Architecture

What does Docker const of?

Docker Container: each process running into it is isolated by system calls (files-system, IPC, kernel interactions, etc)



Building, Handling Images, Lifecycle management of containers, network, storage etc.

Docker daemon API wrapper and usable as CLI

An isolated process

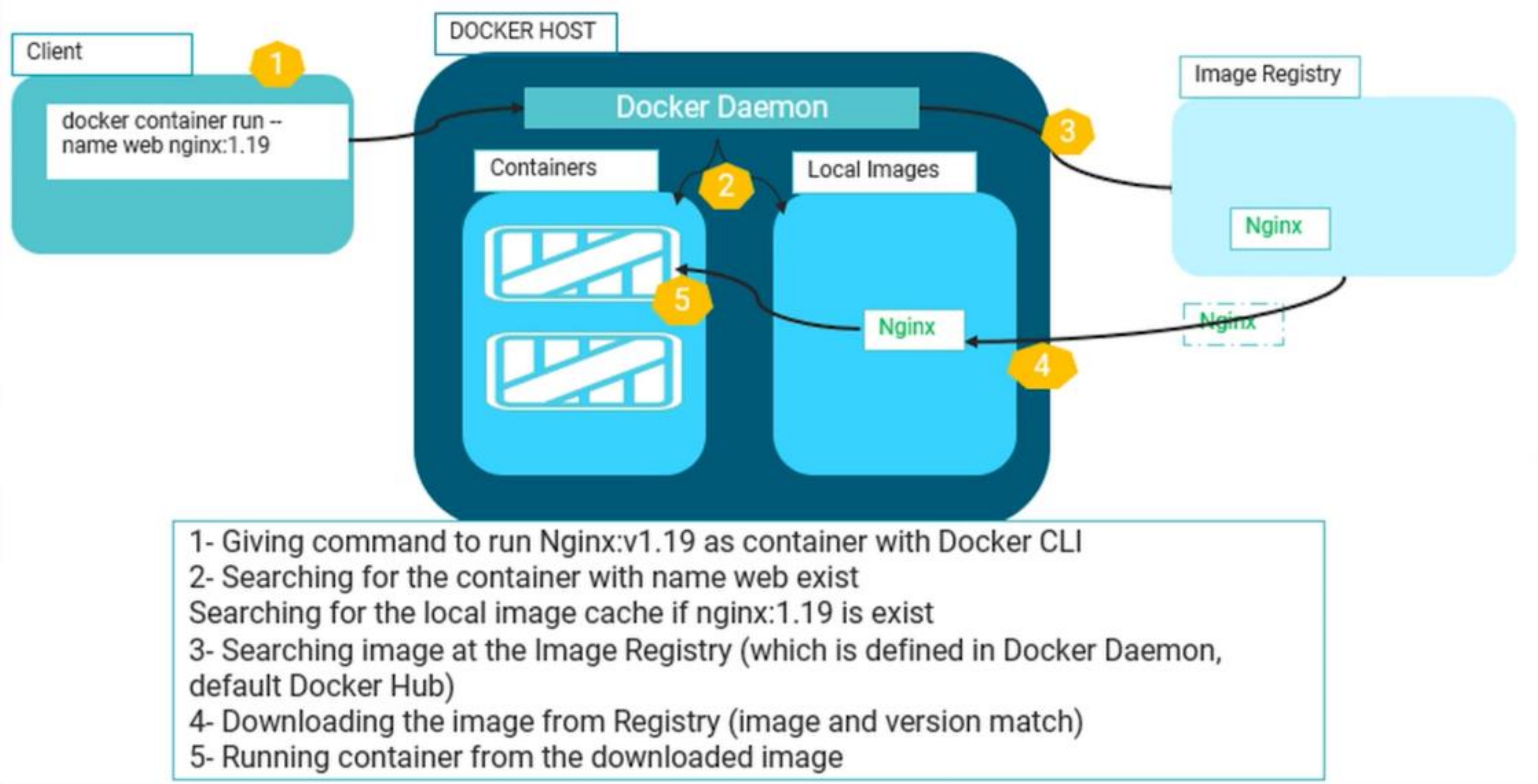
A Library for images

A template and package containing all the necessary items to run as container

G.3.2d Containers: Docker Architecture

How does Docker work?

Docker Container: each one is stored in a repository (Image Registry)



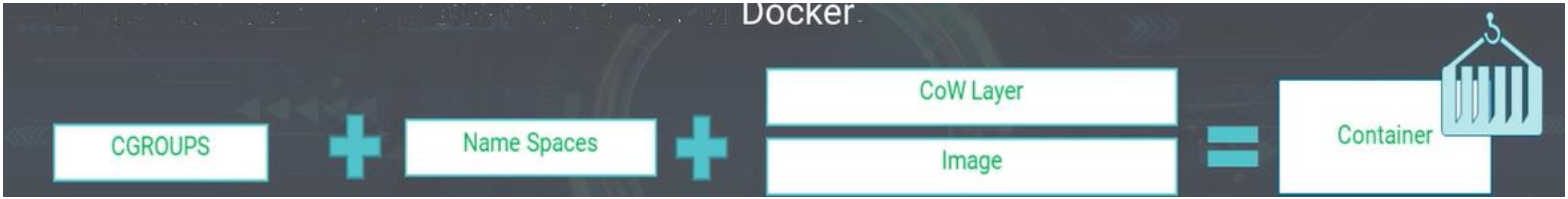
1- Giving command to run Nginx:v1.19 as container with Docker CLI
2- Searching for the container with name web exist
Searching for the local image cache if nginx:1.19 is exist
3- Searching image at the Image Registry (which is defined in Docker Daemon, default Docker Hub)
4- Downloading the image from Registry (image and version match)
5- Running container from the downloaded image

G.3.3a Containers: Docker Management

Docker Containers

Containers: like isolated processes

1. Bring all the items to run the application inside it
2. Status: run, started, stopped, moved, deleted
3. Run-time component of Docker



CGroups:

Kernal Features

Group processes

Control Resources Usage

Allocate HW Resources

1. CPU (CPU set)
2. Memory
3. Disk (Block I/O)

NameSpaces: isolation among Docker containers (introduced in Linux Kernel 2.6.x)

1. PID (process ID)
2. NET
3. IPC (Inter Proc Comms)
4. UTS (Unix Time Share)
5. User namespaces

CoW: Copy on Write. the copy operation is deferred until the first write.

Image: hierarchical tar ball containing:

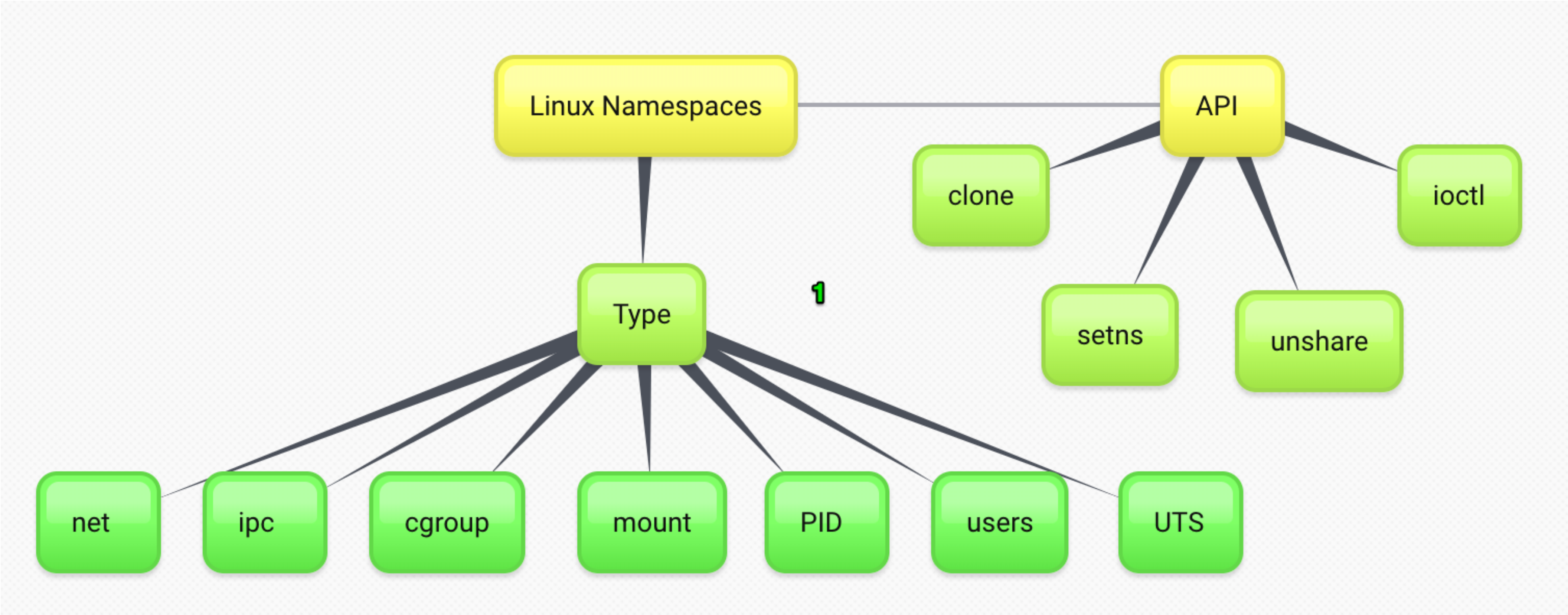
1. O.S.
2. App
3. Lobs
4. Config

G.3.3b Containers: Docker Management

NameSpaces

Namespaces: Linux Kernel Feature

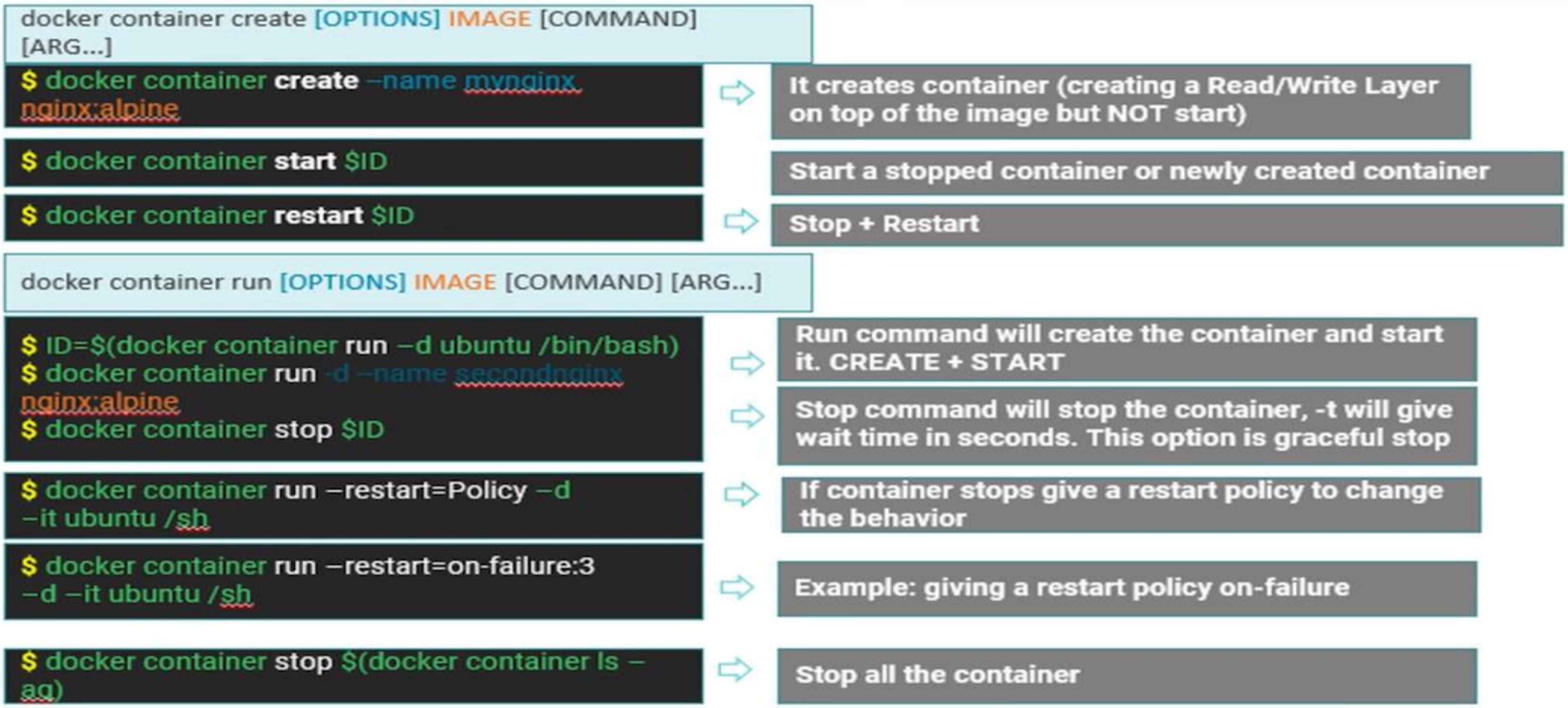
1. wraps a global system resource in an abstraction that makes it appear to the processes within the namespace
2. Changes to the global resource are visible to other processes that are members of the namespace
3. specified as an array of entries inside the namespaces root field



G.3.3c Containers: Docker Management

Commands

Setting Up: Creation/Start/Stop/Run



```
docker container create [OPTIONS] IMAGE [COMMAND] [ARG...]
```

```
$ docker container create –name mynginx nginx:alpine
```

⇨ It creates container (creating a Read/Write Layer on top of the image but NOT start)

```
$ docker container start $ID
```

Start a stopped container or newly created container

```
$ docker container restart $ID
```

⇨ Stop + Restart

```
docker container run [OPTIONS] IMAGE [COMMAND] [ARG...]
```

```
$ ID=$(docker container run –d ubuntu /bin/bash)
$ docker container run -d –name secondnginx nginx:alpine
$ docker container stop $ID
```

⇨ Run command will create the container and start it. CREATE + START

⇨ Stop command will stop the container, -t will give wait time in seconds. This option is graceful stop

```
$ docker container run –restart=Policy –d –it ubuntu /sh
```

⇨ If container stops give a restart policy to change the behavior

```
$ docker container run –restart=on-failure:3 –d –it ubuntu /sh
```

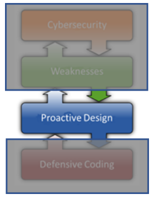
⇨ Example: giving a restart policy on-failure

```
$ docker container stop $(docker container ls –aq)
```

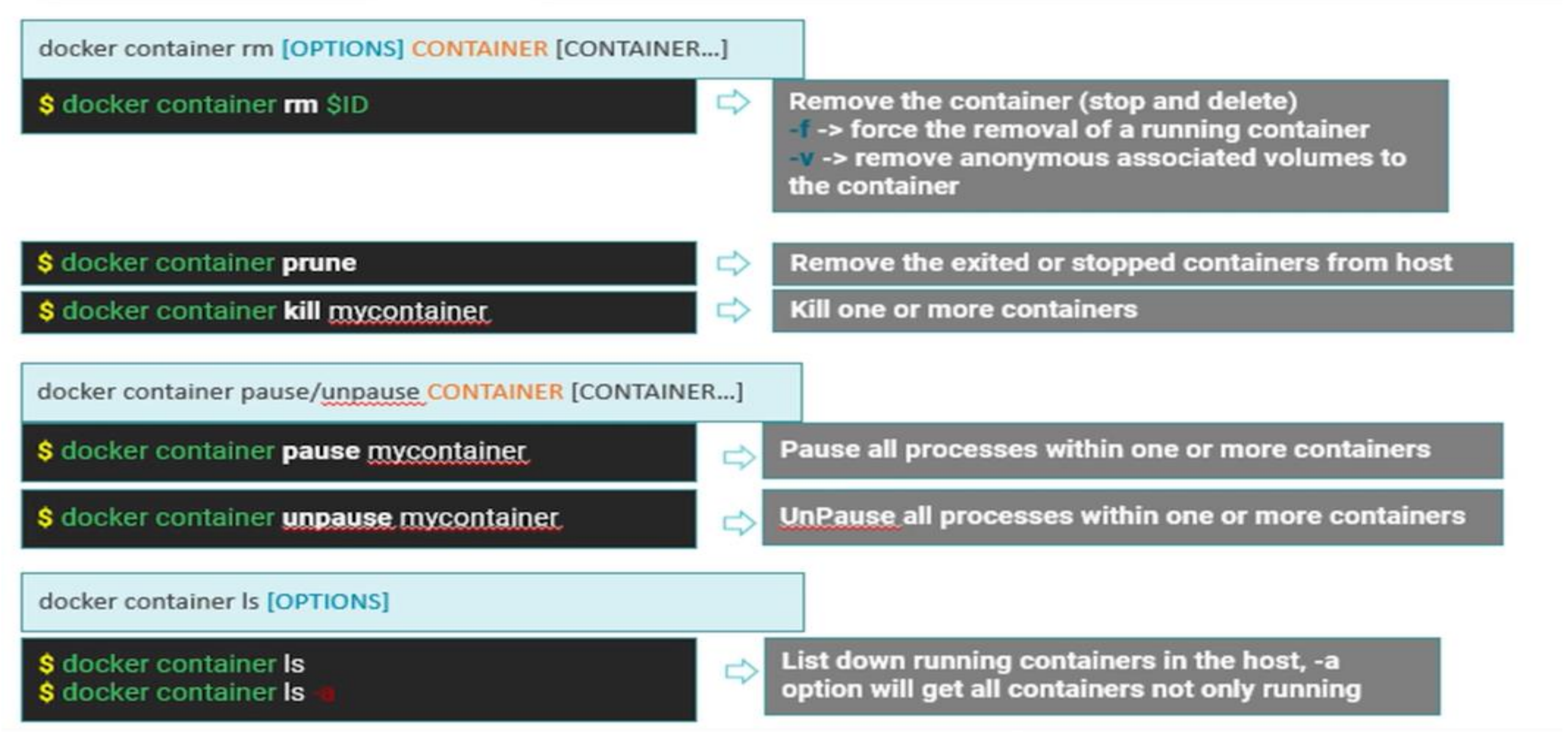
⇨ Stop all the container

G.3.3d Containers: Docker Management

Commands



Smashing Down: rm/prune/kill/pause



```
docker container rm [OPTIONS] CONTAINER [CONTAINER...]
```

```
$ docker container rm $ID
```

Remove the container (stop and delete)
-f -> force the removal of a running container
-v -> remove anonymous associated volumes to the container

```
$ docker container prune
```

Remove the exited or stopped containers from host

```
$ docker container kill mycontainer
```

Kill one or more containers

```
docker container pause/unpause CONTAINER [CONTAINER...]
```

```
$ docker container pause mycontainer
```

Pause all processes within one or more containers

```
$ docker container unpause mycontainer
```

UnPause all processes within one or more containers

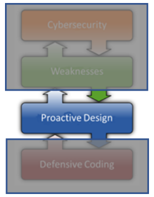
```
docker container ls [OPTIONS]
```

```
$ docker container ls
$ docker container ls -a
```

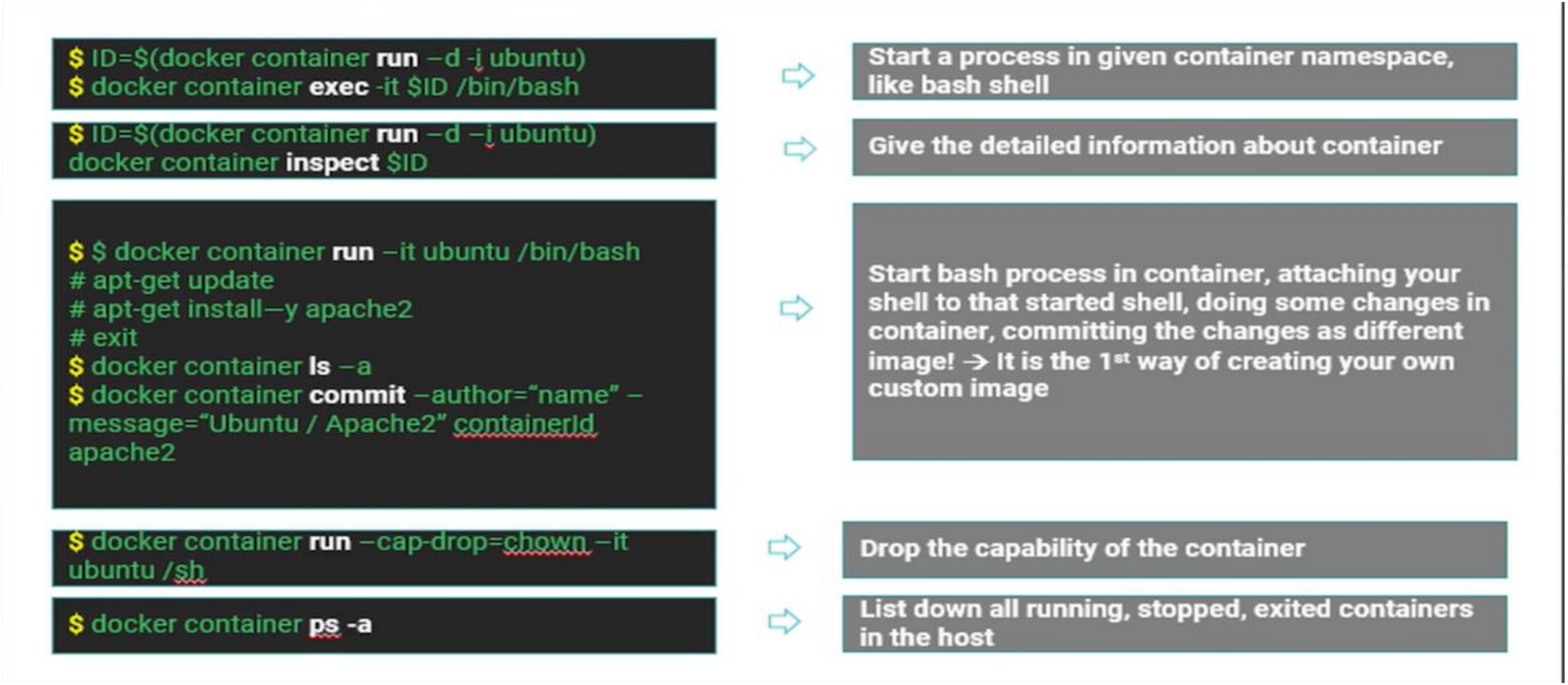
List down running containers in the host, -a option will get all containers not only running

G.3.3e Containers: Docker Management

Commands



Control: exec/inspect/commit/ps



```
$ ID=$(docker container run –d -i ubuntu)
$ docker container exec -it $ID /bin/bash
```

Start a process in given container namespace, like bash shell

```
$ ID=$(docker container run –d –i ubuntu)
docker container inspect $ID
```

Give the detailed information about container

```
$ $ docker container run –it ubuntu /bin/bash
# apt-get update
# apt-get install—y apache2
# exit
$ docker container ls –a
$ docker container commit –author="name" –
message="Ubuntu / Apache2" containerId
apache2
```

Start bash process in container, attaching your shell to that started shell, doing some changes in container, committing the changes as different image! → It is the 1st way of creating your own custom image

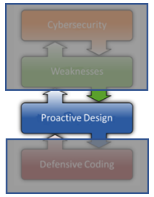
```
$ docker container run –cap-drop=chown –it
ubuntu /sh
```

Drop the capability of the container

```
$ docker container ps -a
```

List down all running, stopped, exited containers in the host

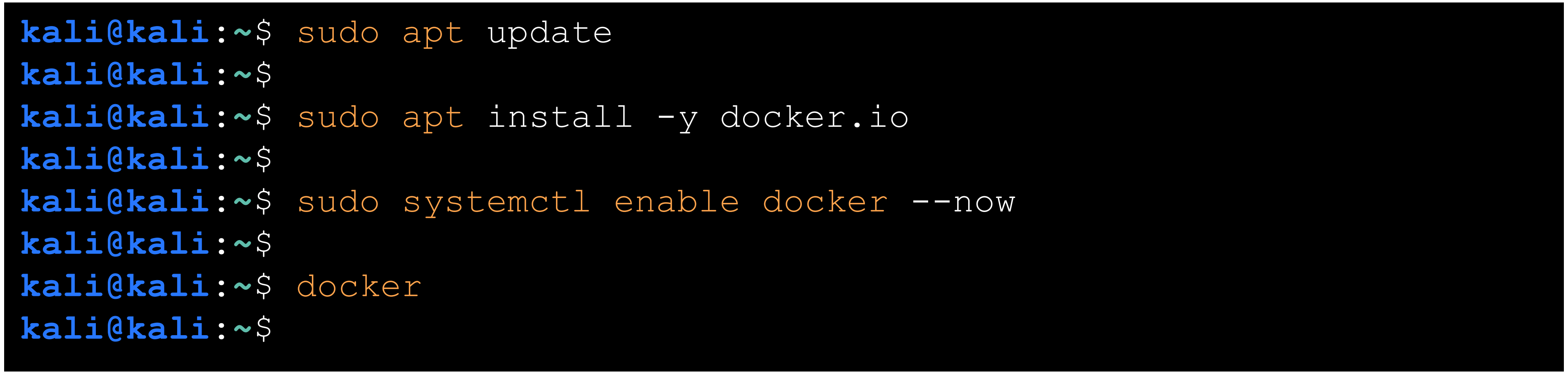
G.3.3e Containers: Docker Management



Lab

Load lab: minute 10.06

Docker installation: https://www.kali.org/docs/containers/installing-docker-on-kali/



```
kali@kali:~$ sudo apt update
kali@kali:~$
kali@kali:~$ sudo apt install -y docker.io
kali@kali:~$
kali@kali:~$ sudo systemctl enable docker --now
kali@kali:~$
kali@kali:~$ docker
kali@kali:~$
```

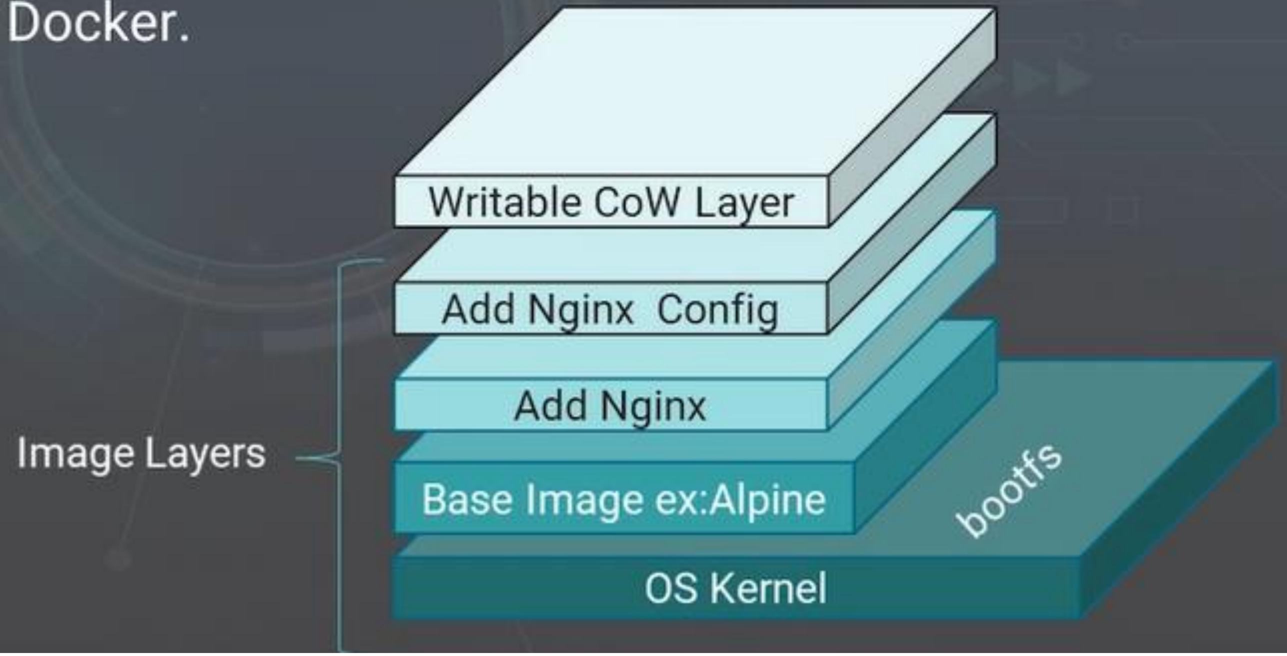
Docker Images: https://www.kali.org/docs/containers/using-kali-docker-images/

https://hub.docker.com/search?q=

G.3.4a Containers: Docker Images and Registry

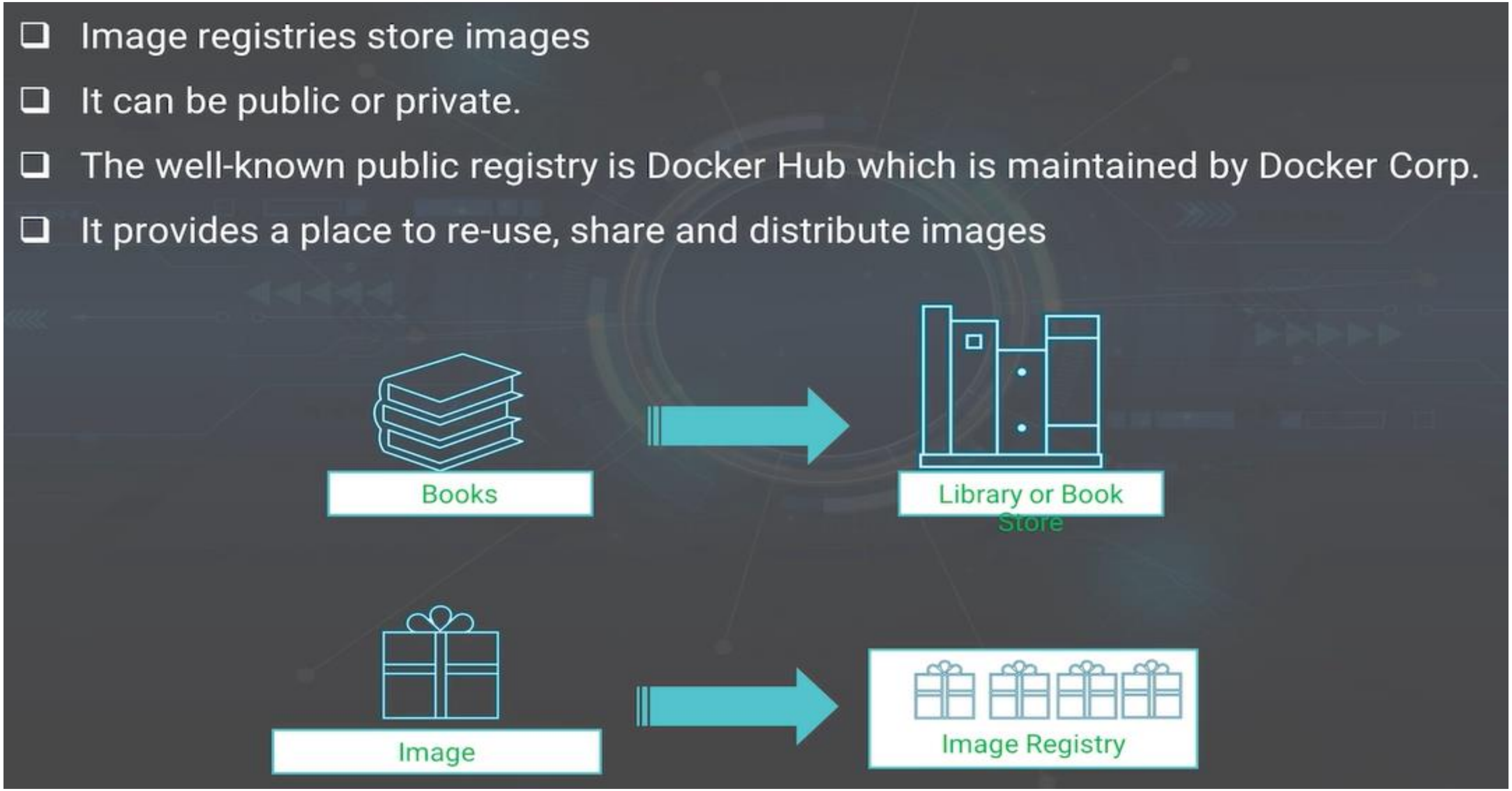
Docker Images

- An image is Read-Only (RO) template.
- Images are used to create containers
- Images are layered, each layer independently loadable, updatable, downloadable with content address hash mechanism.
- Images are built time component of Docker.



G.3.4b Containers: Docker Images and Registry

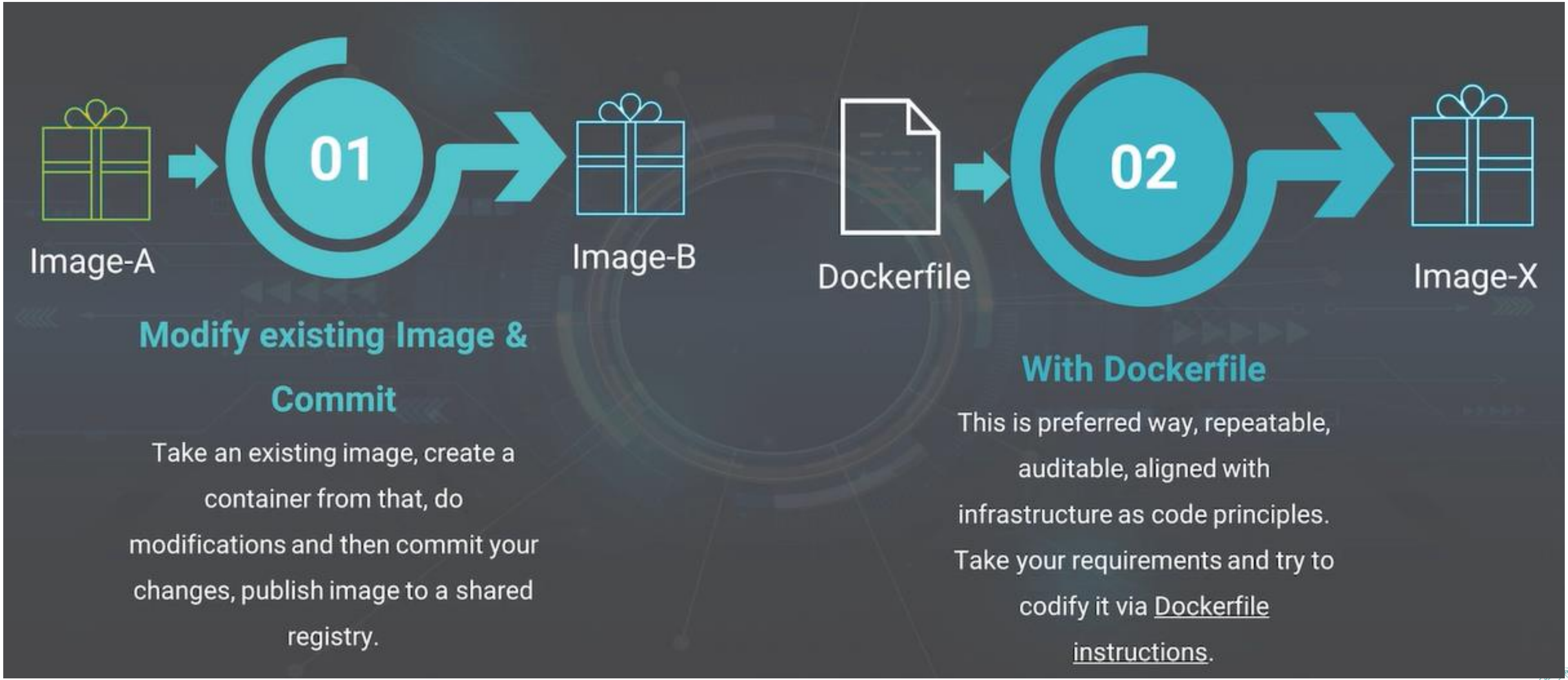
Docker Image Registries



- □ Image registries store images
- □ It can be public or private.
- □ The well-known public registry is Docker Hub which is maintained by Docker Corp.
- □ It provides a place to re-use, share and distribute images

G.3.4c Containers: Docker Images and Registry

How to Create Images



Modify existing Image & Commit

Take an existing image, create a container from that, do modifications and then commit your changes, publish image to a shared registry.

With Dockerfile

This is preferred way, repeatable, auditable, aligned with infrastructure as code principles. Take your requirements and try to codify it via Dockerfile instructions.

G.3.4d Containers: Docker Images and Registry

Docker Image Management 1/2

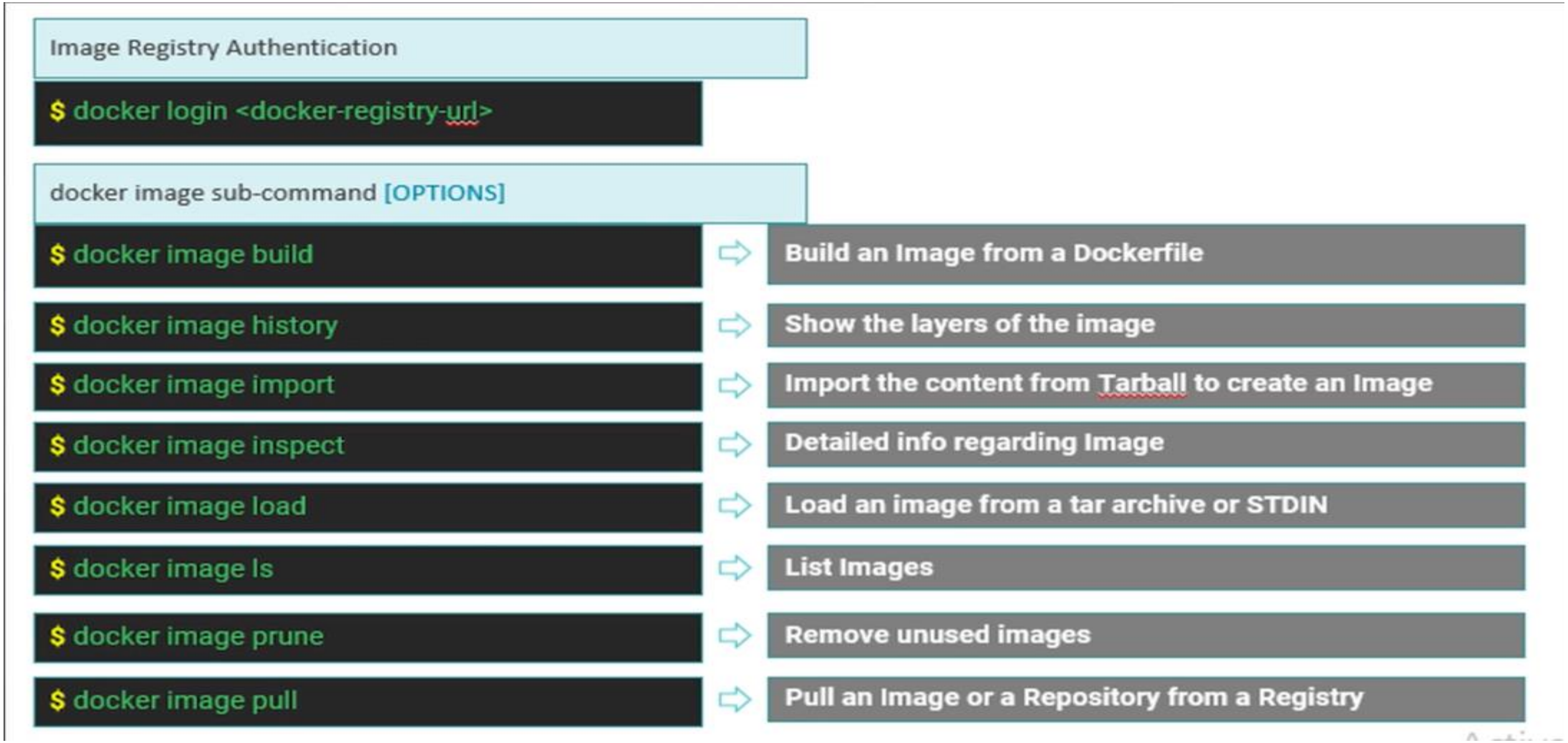


Image Registry Authentication

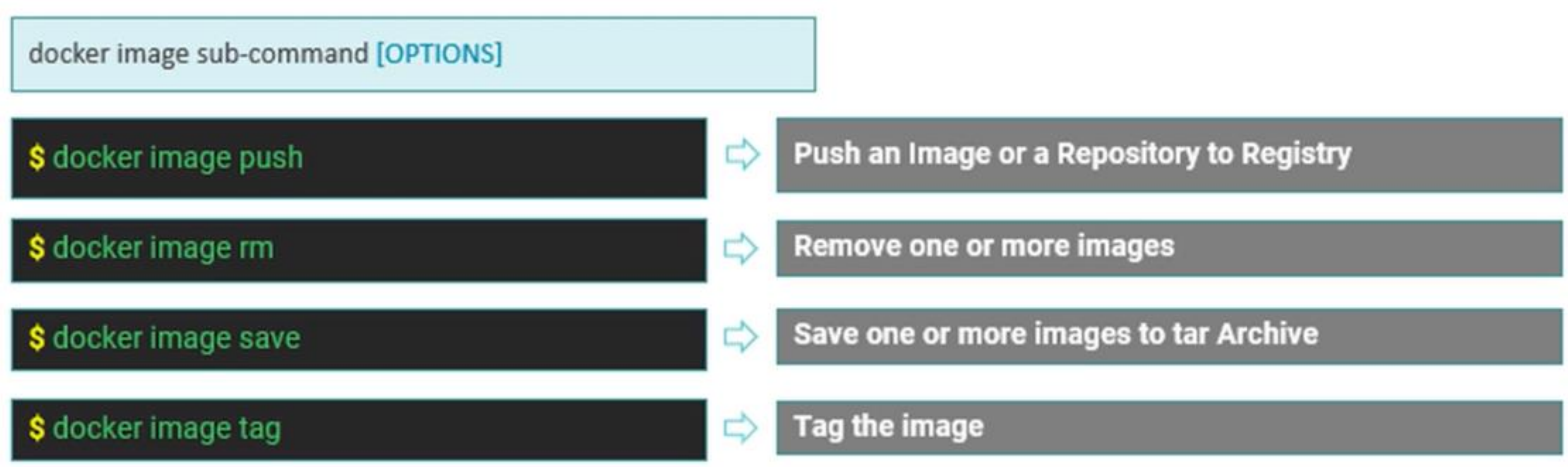
```
$ docker login <docker-registry-url>
```

docker image sub-command [OPTIONS]

`$ docker image build`	Build an Image from a Dockerfile
`$ docker image history`	Show the layers of the image
`$ docker image import`	Import the content from Tarball to create an Image
`$ docker image inspect`	Detailed info regarding Image
`$ docker image load`	Load an image from a tar archive or STDIN
`$ docker image ls`	List Images
`$ docker image prune`	Remove unused images
`$ docker image pull`	Pull an Image or a Repository from a Registry

G.3.4e Containers: Docker Images and Registry

Docker Image Management 2/2

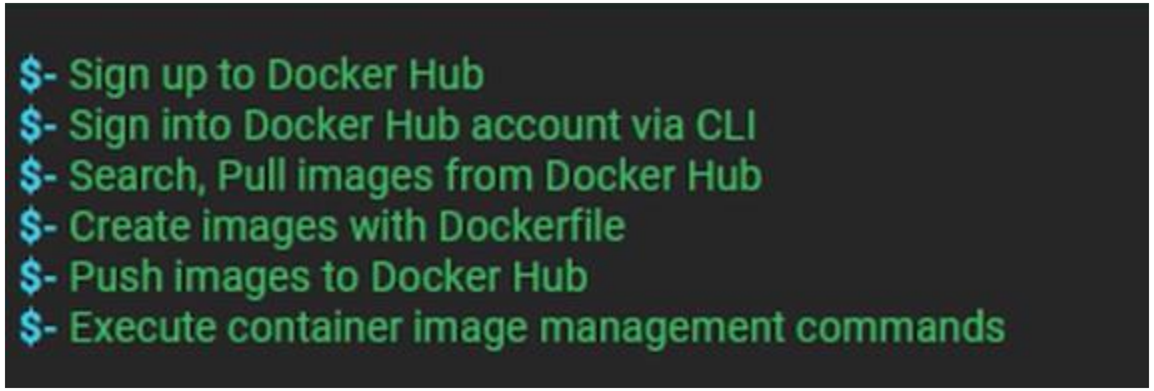


```
docker image sub-command [OPTIONS]
```

`$ docker image push` ⇨	Push an Image or a Repository to Registry
`$ docker image rm` ⇨	Remove one or more images
`$ docker image save` ⇨	Save one or more images to tar Archive
`$ docker image tag` ⇨	Tag the image

G.3.4f Containers: Docker Images and Registry

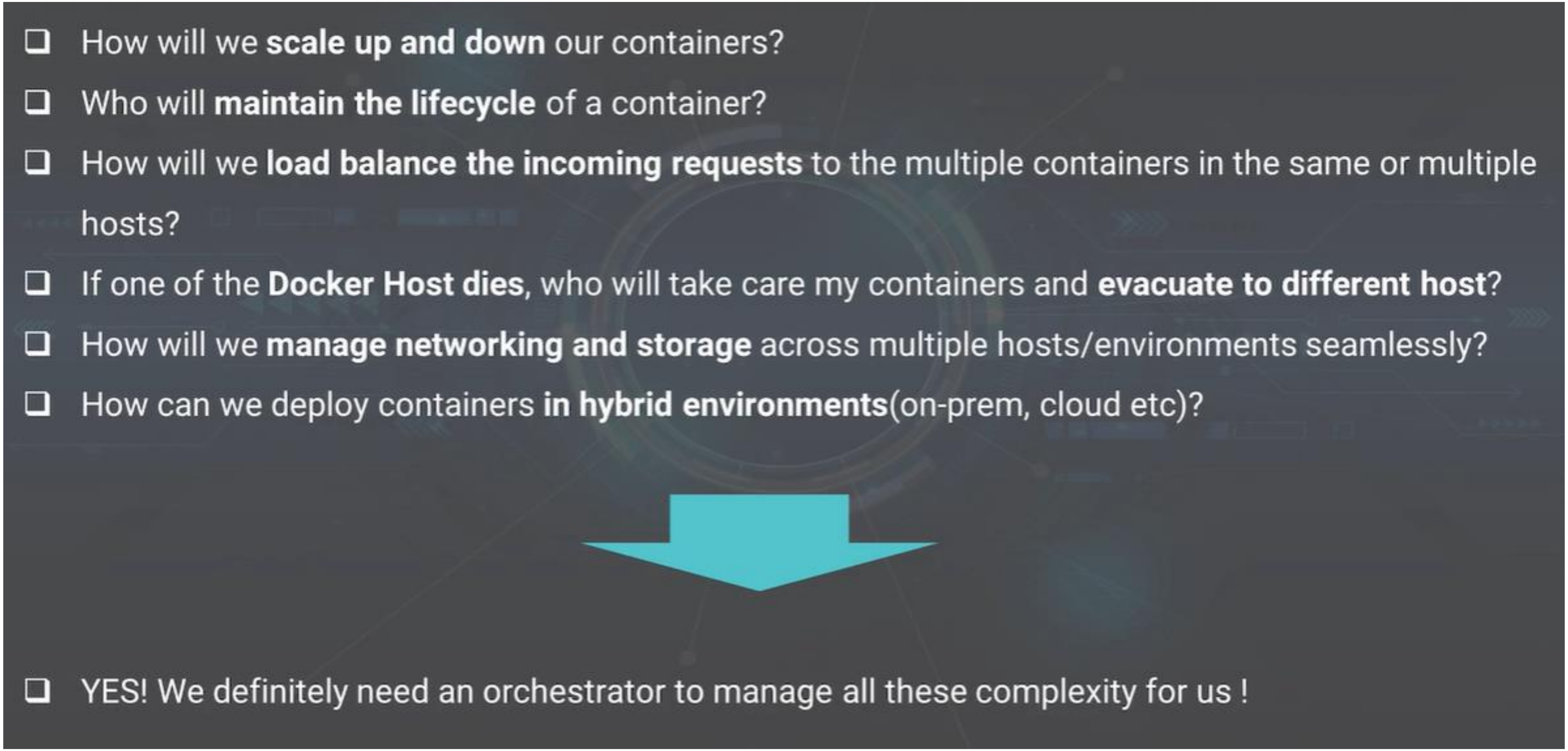
Lab



```
$- Sign up to Docker Hub
$- Sign into Docker Hub account via CLI
$- Search, Pull images from Docker Hub
$- Create images with Dockerfile
$- Push images to Docker Hub
$- Execute container image management commands
```

G.4 Containers: Orchestration

Why we need orchestrator



□ How will we **scale up and down** our containers?

□ Who will **maintain the lifecycle** of a container?

□ How will we **load balance the incoming requests** to the multiple containers in the same or multiple hosts?

□ If one of the **Docker Host dies**, who will take care my containers and **evacuate to different host**?

□ How will we **manage networking and storage** across multiple hosts/environments seamlessly?

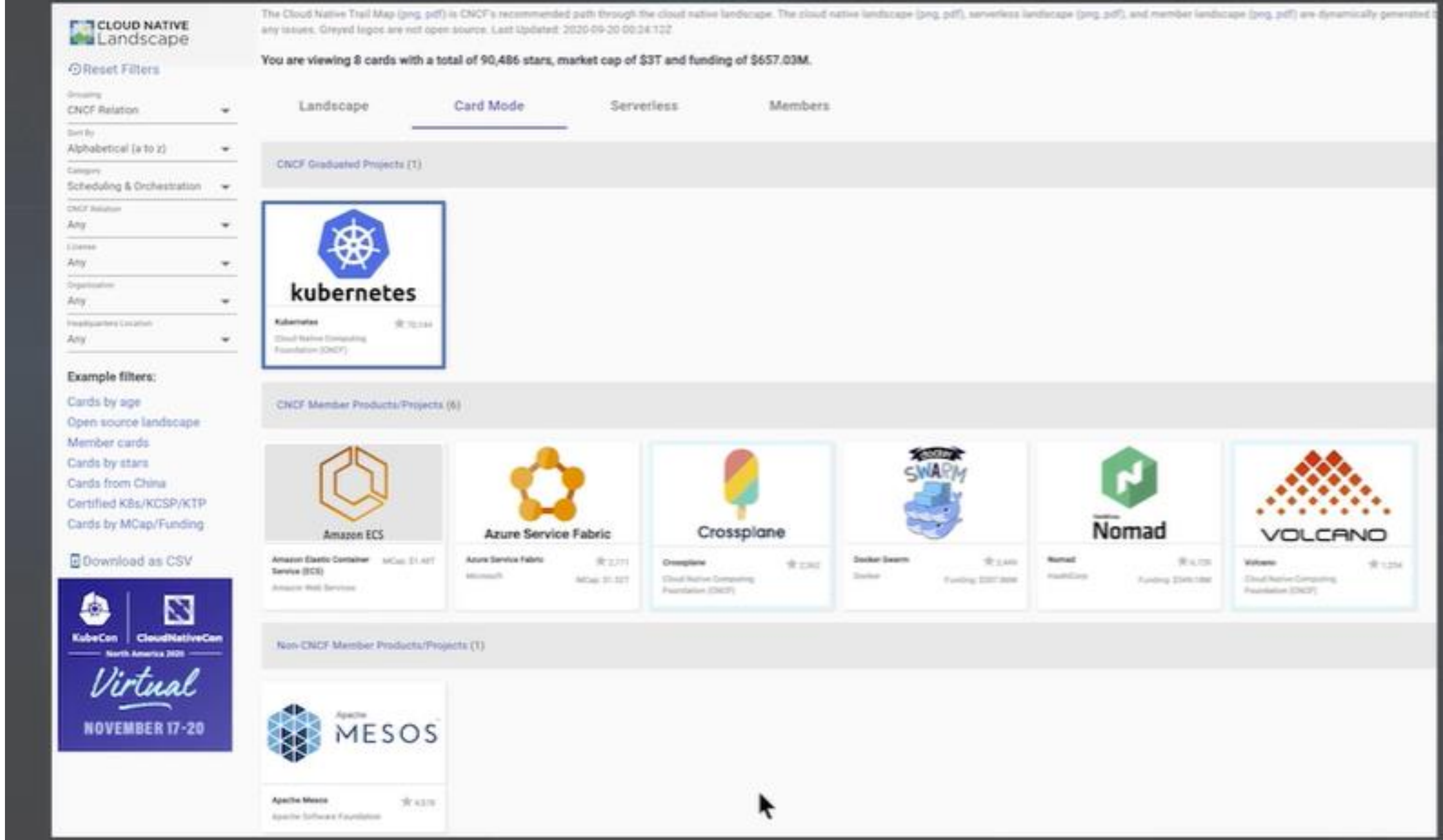
□ How can we deploy containers **in hybrid environments**(on-prem, cloud etc)?

□ YES! We definitely need an orchestrator to manage all these complexity for us !

G.4 Containers: Orchestration

Overview of orchestrator

In the need of a ruler, governor for containers



CLOUD NATIVE Landscape

Reset Filters

Grouping: CNCF Relation

Sort By: Alphabetical (a to z)

Category: Scheduling & Orchestration

CNCF Relation: Any

License: Any

Organization: Any

Headquarters Location: Any

Example filters:

- Cards by age
- Open source landscape
- Member cards
- Cards by stars
- Cards from China
- Certified KBs/KCSP/KTP
- Cards by MCap/Funding

Download as CSV

The Cloud Native Trail Map (png, pdf) is CNCF's recommended path through the cloud native landscape. The cloud native landscape (png, pdf), serverless landscape (png, pdf), and member landscape (png, pdf) are dynamically generated [illegible] any issues. Greyed logos are not open source. Last Updated: 2020-09-20 00:24:12Z

You are viewing 8 cards with a total of 90,486 stars, market cap of $3T and funding of $657.03M.

Landscape | Card Mode | Serverless | Members

CNCF Graduated Projects (1)

- Kubernetes — Cloud Native Computing Foundation (CNCF)

CNCF Member Products/Projects (6)

- Amazon Elastic Container Service (ECS) — Amazon Web Services
- Azure Service Fabric — Microsoft
- Crossplane — Cloud Native Computing Foundation (CNCF)
- Docker Swarm — Docker
- Nomad — HashiCorp
- Volcano — Cloud Native Computing Foundation (CNCF)

Non-CNCF Member Products/Projects (1)

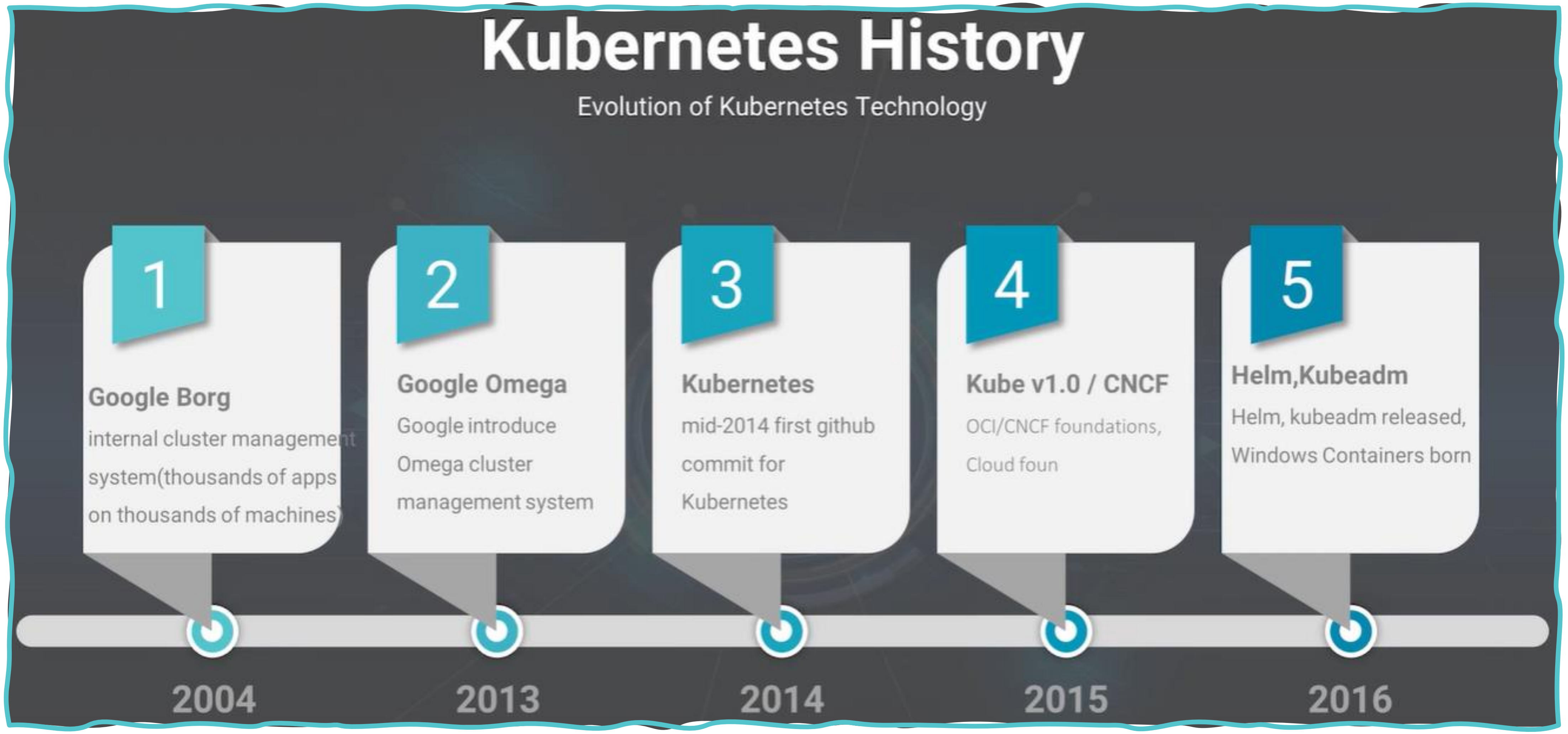
- Apache Mesos — Apache Software Foundation

- ❑ As a PaaS platform
- ❑ Rancher
- ❑ Redhat OpenShift
- ❑ Cloud Foundry Foundation

https://landscape.cncf.io/category=scheduling-orchestration&format=card-mode

G.4 Containers: Orchestration

Overview of orchestrator



Kubernetes History

Evolution of Kubernetes Technology

1 Google Borg
internal cluster management system(thousands of apps on thousands of machines)
2004

2 Google Omega
Google introduce Omega cluster management system
2013

3 Kubernetes
mid-2014 first github commit for Kubernetes
2014

4 Kube v1.0 / CNCF
OCI/CNCF foundations, Cloud foun
2015

5 Helm,Kubeadm
Helm, kubeadm released, Windows Containers born
2016