IONOS Enterprise Cloud

Kubernetes Security Best Practices

Lorenzo Galelli
Snr Cloud Solution Architect

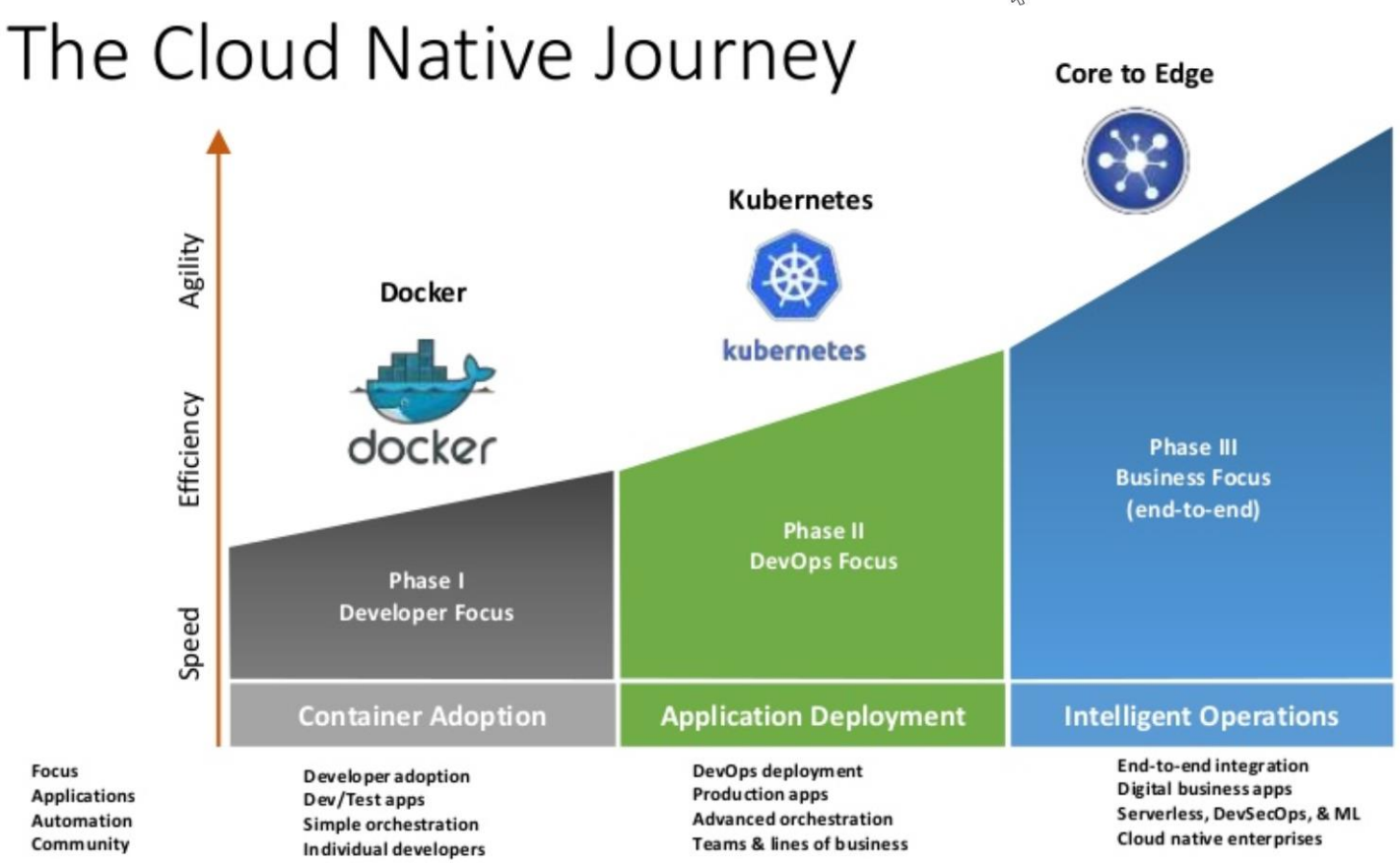
www.ionos.co.uk/enterprise-cloud/

March 2020

IONOS by 1&1

@Virtually_LG
virtuallylg.com

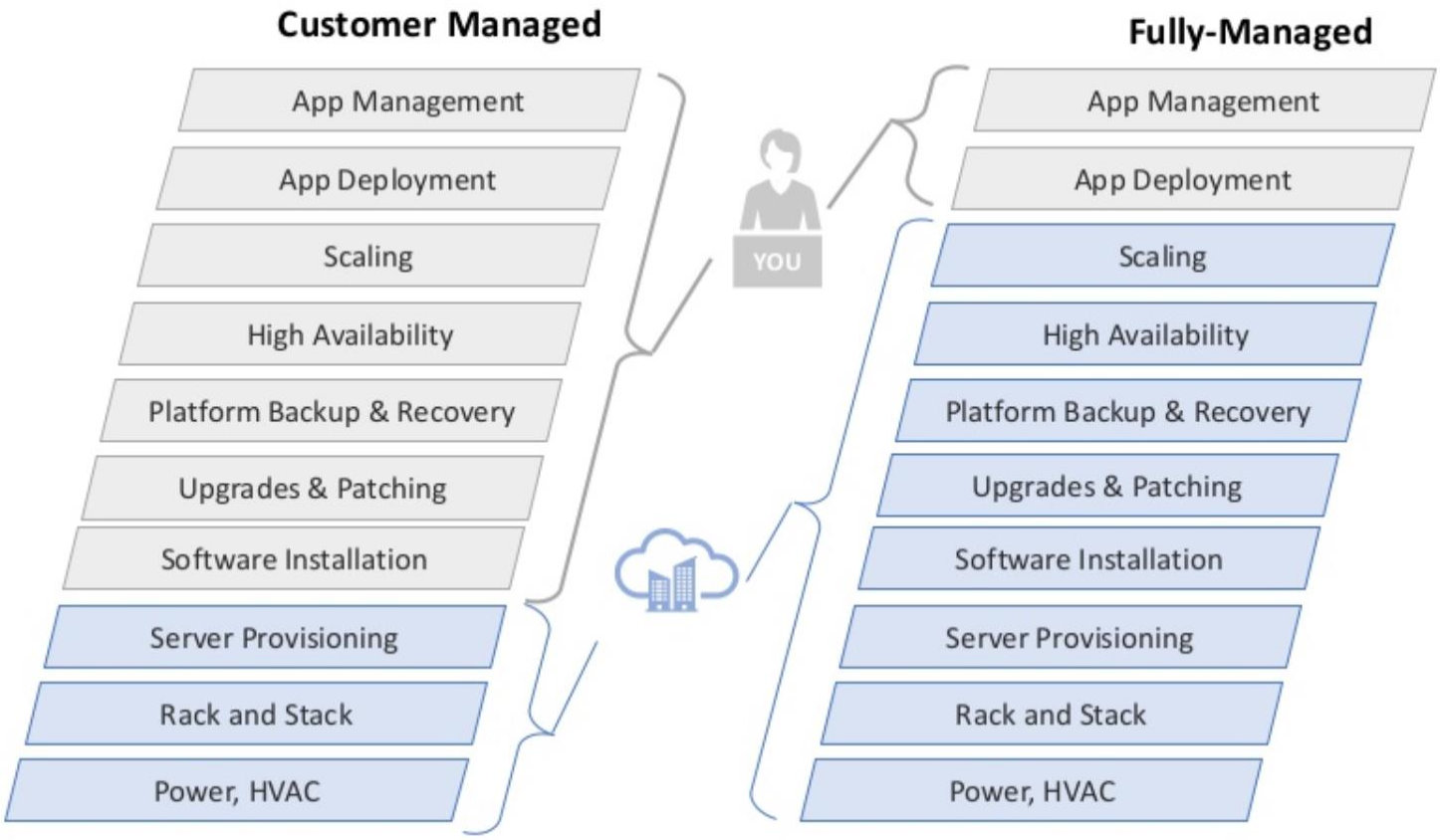
Cloud Native Journey



Kubernetes & Cloud Native Challenges

- Managing, maintaining, upgrading kubernetes control plane
  - API Server, etcd, scheduler etc..
- Managing, maintaining, upgrading kubernetes data plane
  - In place upgrades, deploy parallel cluster etc.
- Figure out container network & storage
  - Overlays, persistent storage etc... - it should just work
- Managing teams
  - How do I manage & control team access to my cluster
- Security, security and security

How are teams addressing complexity & training issues



Benefits

- ✓ Faster Time to Deploy
- ✓ Lower Risk
- ✓ Accelerate Innovation

Which brings us to SECURITY

Common Vulnerabilities and Exposures (CVE)

- runC, the most commonly used low-level container runtime in Docker and Kubernetes environments.
- Vulnerability in runC, which allows host-level code execution breaking out of a running container
  - Discovered and reported by Adam Iwaniuk and Borys Poplawski in early January and published as CVE-2019-5736 on 11 February 2019.
- Significant vulnerability:
  - Enables container isolation breakout with minimal interaction from an authorized host user;
  - Typically allows an attacker to obtain root privileges on the host;
  - Negatively impacts most container environments because many containers run with default Docker security settings and default user (UID 0)

Crypto hacking Tesla's Kube

- Crypto-currency mining using large amounts of computer processing power
- Hackers using systems for crypto-jacking.
- Crypto-jacking becoming more widespread,
- Hackers compromised services offered by Starbucks, YouTube and the UK's Information Commissioner's Office.
- Hackers discovered log-in details to Tesla's Amazon Web Services environment on a Kubernetes console
    - The console was reportedly not password-protected.

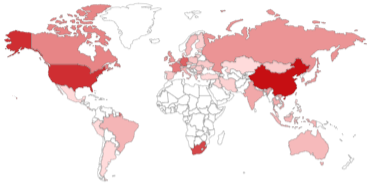
It's not just Tesla

https://www.shodan.io/search?query=etcd,2379:2380

TOTAL RESULTS

2,559

TOP COUNTRIES



China	978
United States	531
South Africa	318
Germany	163
France	82

TOP SERVICES

2379	2,494
Docker	12
HTTP (8080)	9
8081	8
ElasticSearch	6

TOP ORGANIZATIONS

Hangzhou Alibaba Advertisin...	399
Amazon.com	307
Tencent cloud computing	228
Cnservers LLC	223
Google Cloud	84

New Service: Keep track of what you have connected to the Internet. Check out Shodan Monitor

72.162.198.104.bc.googleusercontent.com
Google Cloud
Added on 2020-02-14 07:59:13 GMT
United States

cloud

etcd
Name: cp1
Version: 3.3.3
Uptime: 11h33m59.258462658s
Peers: http://10.240.0.80:2380

Microsoft Azure
Added on 2020-02-14 07:26:15 GMT
India, Chennai

cloud

etcd
Name: openstack-dev
Version: 3.3.12
Uptime: 7481h16m44.181671106s
Peers: http://10.0.12.4:2380

China Telecom
Added on 2020-02-14 07:13:54 GMT
China

etcd
Name: etcd-node1
Version: 3.2.18
Uptime: 1489h22m40.968843017s
Peers: http://192.168.0.114:2380

host116-223-36-89.static.arubacloud.com
ArubaCloud Limited
Added on 2020-02-14 08:16:24 GMT
United Kingdom, Slough

etcd
Name: backend3
Version: 3.3.13
Uptime: 884h45m34.037056358s
Peers: http://backend3:2380

Do you have an upgrade strategy

- CVEs are typically fixed in new minor releases….
- Don’t treat K8s as install once and run forever
- Make your K8s install repeatable and on different versions

- Or use a managed service provider
  - Either automatically upgrade for you and assist through the process

Securing the Control Plane

Enable RBAC with Least Privilege, Disable ABAC, and Monitor Logs

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
   name: example-clusterrolebinding
subjects:
-  kind: User
   name: example-user
   apiGroup: rbac.authorization.k8s.io
roleRef:
   kind: ClusterRole
   name: example-clusterrole
   apiGroup: rbac.authorization.k8s.io
```

Kubernetes.
levels.

cations and services, various developers and/or
es on and on.

ed access control (RBAC).

and access. The RBAC API declares four top-level

ources within a single namespace;
, non-resource endpoints, and namespaced

resources to the Role type;

- **RoleBinding** grants permissions define
- **ClusterRoleBinding** is the same as Rol

- Imperative that every K8s admin understand
- Be aware of ABAC authorization method, doe
  anymore

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
   name: example-clusterrole
rules:
-  apiGroups: [""]
   resources: ["pods"]
   verbs: ["get", "watch", "list"]
```

Keeping Control Plane Fit and Healthy

- (Transport Layer Security) TLS Everywhere
  - TLS should be enabled for every component that supports it to prevent traffic sniffing,
  - Verify the identity of the server, and (for mutual TLS) verify the identity of the client.
- Upgrade to the Latest Version K8s where possible.
  - New security features, not just bug fixes
  - Added in every quarterly update
- Upgrades and support become more difficult the further behind you fall,
  - Plan to upgrade at least once per quarter
  - Using a managed Kubernetes provider can make upgrades very easy.

Namespace is your friend

- Use Namespaces to Establish Security Boundaries
- Creating separate namespaces
  - Important first level of isolation between components.
  - Easier to apply security controls such as Network Policies when different types of workloads are deployed in separate namespaces.
- Is your team using namespaces effectively? Find out now by checking for any non-default namespaces:

```
kubectl get ns
NAME          STATUS    AGE
default       Active    16m
kube-public   Active    16m
kube-system   Active    16m
```

Harden Node Security

Follow these three steps to improve the security posture on your nodes:

- Ensure the host is secure and configured correctly.
  - Check your configuration against CIS Benchmarks; many products feature an autochecker that will assess conformance with these standards automatically.
- Control network access to sensitive ports.
  - Make sure that your network blocks access to ports used by kubelet, including 10250 and 10255.
  - Consider limiting access to the Kubernetes API server except from trusted networks.
  - Avoid being the next News headline.
- Minimize administrative access to Kubernetes nodes.
  - Access to the nodes in your cluster should generally be restricted — debugging and other tasks can usually be handled without direct access to the node.
- Look to use tools like kube-bench, kube-audit, kube-hunter and kubesec

Enable Audit Logging

- Turn on Audit Logg
- Make sure you have
  - Especially any
  - Log entries wil
    that an attacke
- Read requests such
  - Response obje
  - Requests invo
    exported.
  - All other reque
- Keeping these logs
  - Should be tran
    breach.
- Use an encryption p

```
{
 "kind":"Event",
 "apiVersion":"audit.k8s.io/v1beta1",
 "metadata":{"creationTimestamp":"2018-03-21T21:47:07Z"},
 "level":"Metadata",
 "timestamp":"2018-03-21T21:47:07Z",
 "auditID":"20ac14d3-1214-42b8-af3c-31454f6d7dfb",
 "stage":"RequestReceived",
"requestURI":"/api/v1/namespaces/default/persistentvolumeclaims"
,
 "verb":"list",
 "user": {
  "username":"lorenzo.galelli@example.org",
  "groups":[ "system:authenticated" ]
 },
 "sourceIPs":[ "77.68.66.233" ],
 "objectRef": {
  "resource":"persistentvolumeclaims",
  "namespace":"default",
  "apiVersion":"v1"
 },
 "requestReceivedTimestamp":"2020-03-02T21:47:07.603214Z",
 "stageTimestamp":"2020-03-02T21:47:07.603214Z"
```

or anomalous or

tion failures could mean

s saved in the audit logs

ap, only metadata is

ed in audit logs.

ompromise.

ng in the event of a

and not for config maps.

Securing the Workloads

Containers are numerous and everywhere

- Although the microservices design pattern that powers containerized applications underlies many of the benefits of containerization, it also creates security blind spots and increases your attack surface. As more containers are deployed, maintaining adequate visibility into your critical cloud-native infrastructure components becomes more difficult. The distributed nature of containerized apps makes it difficult to quickly investigate which containers might have a newly discovered zero-day vulnerability, which ones are running as *privileged*, or other factors.

Pod Security

- Isolate your pods
  - Apply at least one network policy to every pod to ensure isolation
  - Remove the related application to both lateral and north-south threats.
  - **Apply a “deny all” policy to all pods as a default first step**.
- Explicitly allow Internet access for pods using labels that need it
  - Associate the labels with the pods that require Internet reachability
  - Create a network policy that targets those labels.
- Explicitly allow necessary pod-to-pod communication
  - Limit security risk by allowing pods within the same namespace to freely communicate with each other.
- Separate Sensitive Workloads
  - Run sensitive workloads on a dedicated set of machines.
  - Reduces the risk of a sensitive application being accessed through a less-secure application that shares a container runtime or host. ies to steal them.

Admission Controllers

- Prevent risky configuration with PodSecurityPolicy
  - Arguably the most important one from a security perspective.
  - Defines a set of conditions a pod must run with to be accepted in the system.
  - Pod security policies can be used to prevent containers from running as root or to make sure the container's root filesystem is mounted read-only.
  - Think of PodSecurityPolicy as authorization for Pods
- Run a Cluster-wide Pod Security Policy
  - Pod Security Policy sets defaults for how workloads are allowed to run in your cluster.
  - Consider defining a policy and enabling the Pod Security Policy admission controller..
- Enforce image registry governance
  - Admission controllers can be used to allow images to be pulled from trusted registries while denying all untrusted image registries.
- Ensure adherence to good DevOps practices
  - Admission controllers can be used to enforce internal policies for use of labels on various objects or consistently adding annotations to objects.

Images are numerous and everywhere

Image Security

- Know your base image when building containers
  - When in doubt stick to official images
  - Or start from a sane built image like alpine linux
- Smaller the image the better
  - Less surface to attack
  - Quicker to push, quicker to pull
- Don’t rely on :latest TAG
  - Latest image today might not be the latest tomorrow
- Know the specific version you are working with
  - If there is a new vulnerability announced for OS version x.1.0 you will have immediate visibility
- Check for vulnerabilities regularly
- Be wary of images that require broad access to paths on the host

What else can you be doing, Can you answer these questions confidently

- Where are images used in containers coming from?
- How long ago were the images scanned for vulnerabilities?
- Which of your containers are impacted by known vulnerabilities, and what's their severity?
- Are any of these containers in production impacted by a known vulnerability?
- Which vulnerable running containers or deployments should be prioritize first for remediation?
- Which deployments are using privileged containers, meaning they have full access to the host operating system?
- What container applications services are exposed outside of the Kubernetes cluster?
- Can we tell which processes are running in any container in any cluster?
- Which network communication pathways are active but are not being used in production?
- Which running deployment have had an adversary attempt to run a specific runtime exploit?
- What team in the organization owns a particular running application?

Thank You!

Lorenzo Galelli

Senior Cloud Solution Architect

Lorenzo.Galelli@cloud.ionos.co.uk