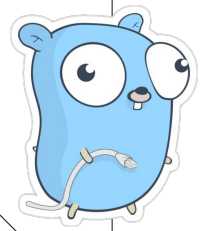

KubeCon
CloudNativeCon
Europe 2023
TiKV


# How to Make Your K8s Cluster Survive When It Has No Internet Access

*Christophe Jauffret, Nutanix*

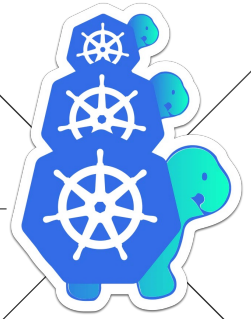
~ > whoami



NUTANIX

# Christophe Jauffret

*Staff Solutions Architect @ Nutanix*



kubernetes AND CLOUD NATIVE ASSOCIATE

# Let’s start with a container

KubeCon CloudNativeCon Europe 2023

```
 ~ ▶ docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
fcdb9667c46b: Downloading [=========================>                         ]  15.54MB/30.06MB
d9f3ddb89cd9: Downloading [================>                                  ]   8.37MB/25.39MB
7bae30547136: Download complete
3cf7afb4b643: Waiting
e20f1c808987: Waiting
e40c7e401531: Waiting
```

# Let’s start with a container

KubeCon CloudNativeCon Europe 2023

```
 ~ ▶ docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
fcdb9667c46b: Pull complete
d9f3ddb89cd9: Pull complete
7bae30547136: Pull complete
3cf7afb4b643: Pull complete
e20f1c808987: Pull complete
e40c7e401531: Pull complete
Digest: sha256:f4e3b6489888647ce1834b601c6c06b9f8c03dee6e097e13ed3e28c01ea3ac8c
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
 ~ ▶
```

# And a container in K8s



```
 ~ ▶ kubectl create deployment my-nginx --image=nginx
deployment.apps/my-nginx created
```

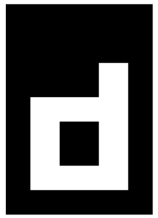
# And a container in K8s



```
 ~ ▶ kubectl get event
LAST SEEN   TYPE     REASON              OBJECT                      MESSAGE
3s          Normal   Scheduled           pod/my-nginx-7cddc-n88lm    Successfully assigned default/my-nginx-7cddc5685c-n88lm to node-0
3s          Normal   Pulling             pod/my-nginx-7cddc-n88lm    Pulling image "nginx"
2s          Normal   Pulled              pod/my-nginx-7cddc-n88lm    Successfully pulled image "nginx" in 4s
2s          Normal   Created             pod/my-nginx-7cddc-n88lm    Created container nginx
2s          Normal   Started             pod/my-nginx-7cddc-n88lm    Started container nginx
4s          Normal   SuccessfulCreate    replicaset/my-nginx-7cddc   Created pod: my-nginx-7cddc5685c-n88lm
4s          Normal   ScalingReplicaSet   deployment/my-nginx         Scaled up replica set my-nginx-7cddc5685c to 1
 ~ ▶
```

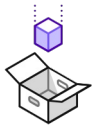
# Containers need internet



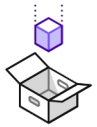


Container runtime

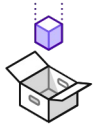
Images pull



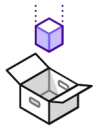
mongo



nginx



redis



kube-vip

registry.k8s.io

docker.io

ghcr.io

quay.io

# Internet is everywhere




*Mika Baumeister on Unsplash*

# Enterprise life is different



```
Events:
  Type     Reason     Age                From               Message
  ----     ------     ----               ----               -------
  Normal   Scheduled  2m6s               default-scheduler  Successfully assigned kube-system/cilium-zqwqn to test-kcp-vxvzw
  Warning  Failed     66s                kubelet            Failed to pull image
"quay.io/cilium/cilium:v1.13.1@sha256:428a09552707cc90228b7ff48c6e7a33dc0a97fe1dd93311ca672834be25beda": rpc error: code = DeadlineExceeded
desc = failed to pull and unpack image "quay.io/cilium/cilium@sha256:428a09552707cc90228b7ff48c6e7a33dc0a97fe1dd93311ca672834be25beda":
failed to resolve reference "quay.io/cilium/cilium@sha256:428a09552707cc90228b7ff48c6e7a33dc0a97fe1dd93311ca672834be25beda": failed to do
request: Head "https://quay.io/v2/cilium/cilium/manifests/sha256:428a09552707cc90228b7ff48c6e7a33dc0a97fe1dd93
311ca672834be25beda": dial tcp 54.85.89.0:443: i/o timeout
  Warning  Failed     66s                kubelet            Error: ErrImagePull
  Normal   BackOff    65s                kubelet            Back-off pulling image
"quay.io/cilium/cilium:v1.13.1@sha256:428a09552707cc90228b7ff48c6e7a33dc0a97fe1dd93311ca672834be25beda"
  Warning  Failed     65s                kubelet            Error: ImagePullBackOff
  Normal   Pulling    51s (x2 over 2m6s)  kubelet           Pulling image
"quay.io/cilium/cilium:v1.13.1@sha256:428a09552707cc90228b7ff48c6e7a33dc0a97fe1dd93311ca672834be25beda"
```

# Enterprise life is different



## Limited Network

- Security rules
- Destination filtering
- Flow inspection
- Network Proxy
- User Authentication
- DMZ

**Common Security Posture**

# Limited network: Security rules



- You need to identify external images/components needed and configure your firewall accordingly
  - Part of the install tools documentation

**Firewall Requirements**

NKE only supports HTTP unauthenticated proxy. Use the IP or the fully qualified domain name (FQDN) format. Ensure that your firewall allows NKE VMs and CVMs to reach the below domains and subdomains. Also, exclude the following domains from the security-sockets layer (SSL) inspection in the firewall.

- `*.cloudfront.net`
- `*.quay.io`
- `ntnx-portal.s3.amazonaws.com`
- `portal.nutanix.com`
- `release-api.nutanix.com`
- `s3*.amazonaws.com`
- `*.compute-1.amazonaws.com`

# Limited network: Security rules



- CLI to retrieve it

Kubeadm

```
~ ▶ kubeadm config images list
registry.k8s.io/kube-apiserver:v1.26.3
registry.k8s.io/kube-controller-manager:v1.26.3
registry.k8s.io/kube-scheduler:v1.26.3
registry.k8s.io/kube-proxy:v1.26.3
registry.k8s.io/pause:3.9
registry.k8s.io/etcd:3.5.6-0
registry.k8s.io/coredns/coredns:v1.9.3
```

Cluster API

```
~ ▶ clusterctl init list-images -i nutanix -b kubeadm -c kubeadm --core cluster-api
gcr.io/kubebuilder/kube-rbac-proxy:v0.8.0
ghcr.io/nutanix-cloud-native/cluster-api-provider-nutanix/controller:v1.1.3
registry.k8s.io/cluster-api/cluster-api-controller:v1.4.1
registry.k8s.io/cluster-api/kubeadm-bootstrap-controller:v1.4.1
registry.k8s.io/cluster-api/kubeadm-control-plane-controller:v1.4.1
```

# Limited network: Proxy



- Proxy must be configured at your container runtime level to allow it to get container images

- It could be a setting in your k8s distribution, or set with environment variables:

**HTTP_PROXY**
**HTTPS_PROXY**
**NO_PROXY**

# Limited network: Proxy & containerd



```
~ ▶ cat /etc/systemd/system/containerd.service.d/http-proxy.conf
[Service]
Environment="HTTP_PROXY=http://<proxy>:3128"
Environment="HTTPS_PROXY=http://<proxy>:3128"
Environment="NO_PROXY=<localdomain>"
EOF


~ ▶ sudo systemctl daemon-reload
~ ▶ sudo systemctl restart containerd

~ ▶ systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/etc/systemd/system/containerd.service; enabled; vendor preset: enabled)
    Drop-In: /etc/systemd/system/containerd.service.d
             └─max-tasks.conf, memory-pressure.conf, http-proxy.conf
     Active: active (running) since Thu 2023-03-30 10:05:35 UTC; 1min 45s ago
       Docs: https://containerd.io
   Main PID: 1021 (containerd)
      Tasks: 23
     Memory: 718.2M
        CPU: 10.553s
     CGroup: /system.slice/containerd.service
             ├─1021 /usr/local/bin/containerd
             └─1209 /usr/local/bin/containerd-shim-runc-v2 -namespace k8s.io -id XXX -address /run/containerd/containerd.sock
```

# Limited network: Proxy for apps



- Runtime is now using the proxy, but not your apps
- Proxy has to be configured within the pod if internet access is needed
- It’s pretty common to use the same environment variables in pod configuration (sometime lowercase)
- Be careful to disable proxy for internal k8s services

```
spec:
  containers:
  - env:
    - name: HTTP_PROXY
      value: http://<proxy>:3128
    - name: HTTPS_PROXY
      value: http://<proxy>:3128
    - name: NO_PROXY
      value: localhost,<locadomain>
    image: nginx
    imagePullPolicy: Always
    name: web-apps
```

# Limited network: Dynamic Proxy for apps



- Kyverno: policy engine designed for Kubernetes
  - Kyverno policies can **validate, mutate, generate, and clean-up** Kubernetes resources
  - Policies are Kubernetes resources, no new language is required to write them.

```
# Add the Helm repository
helm repo add kyverno https://kyverno.github.io/kyverno/

# Scan your Helm repositories to fetch the latest available charts.
helm repo update

# Install the Kyverno Helm chart into a new namespace called "kyverno"
helm install kyverno kyverno/kyverno -n kyverno --create-namespace
```


- Use Kyverno to automatically configure Proxy for apps

# Limited network: Dynamic Proxy for apps



```yaml
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: add-pod-proxies
spec:
  rules:
  - name: add-pod-proxies
    match:
      any:
      - resources:
          kinds:
          - Pod
    mutate:
      patchStrategicMerge:
        spec:
          containers:
            - (name): "*"
              env:
              - name: HTTP_PROXY
                value: http://<proxy>:3128
              - name: HTTPS_PROXY
                value: http://<proxy>:3128
              - name: NO_PROXY
                value: localhost,<locadomain>
```

# Limited network: Dynamic Proxy for apps

KubeCon CloudNativeCon Europe 2023

```yaml
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: add-pod-proxies
spec:
  rules:
  - name: add-pod-proxies
    match:
      all:
      - resources:
          kinds:
          - Pod
          selector:
            matchLabels:
              proxy: inject
    mutate:
      patchStrategicMerge:
        spec:
          containers:
            - (name): "*"
              env:
              - name: HTTP_PROXY
                value: http://<proxy>:3128
              - name: HTTPS_PROXY
                value: http://<proxy>:3128
              - name: NO_PROXY
                value: localhost,<locadomain>
```

# Enterprise life is different



## Limited Network

- Security rules
- Destination filtering
- Flow inspection
- Network Proxy
- User Authentication
- DMZ

**Common Security Posture**

## Isolated Network

- No internet access
- Bastion
- Recorded session

**High Security Posture**

# Isolated network



- Search in documentation

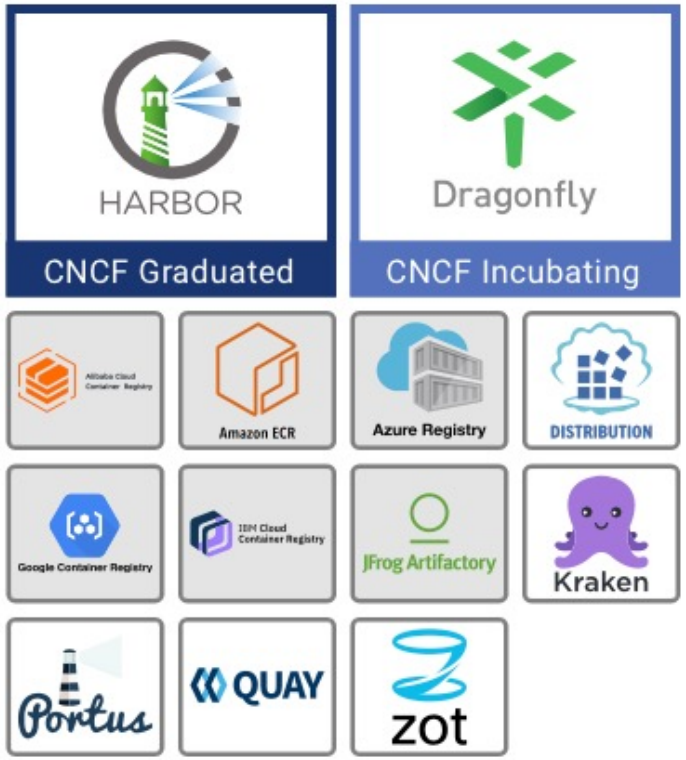
**airgap or no internet connection install method**

- Logic is often the same
  - Identify the required container images/artifacts
  - Pre-pull these items
  - Store them locally
  - Configure the installer/runtime to use these local images/artifacts

# Isolated network: local registry



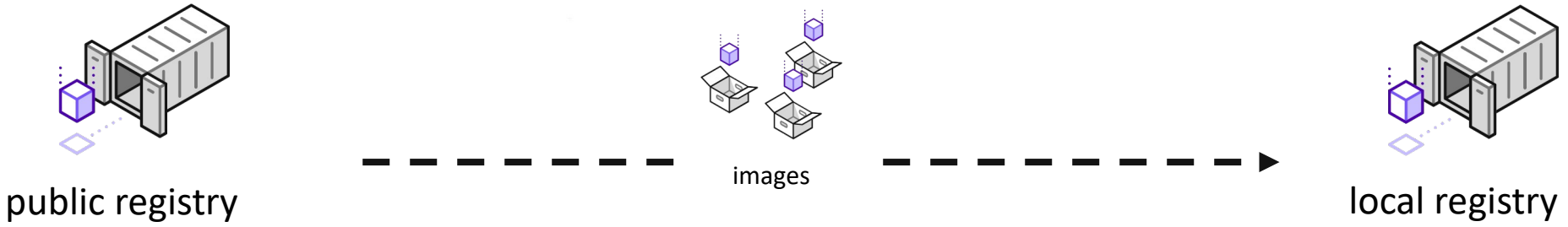
- Having a **local registry** is an absolute requirement
- CNCF landscape offers a lot of options
- Some features can be really useful in case of complex/no internet access:
  - Automatic Replication across registries
  - Security and vulnerability analysis
  - Content signing and validation
  - Image lifecycle
  - P2P image distribution
  - Proxy cache



# Local registry with Internet Access



- Use registry image replication feature




Replications

+ NEW REPLICATION RULE | REPLICATE | ACTIONS

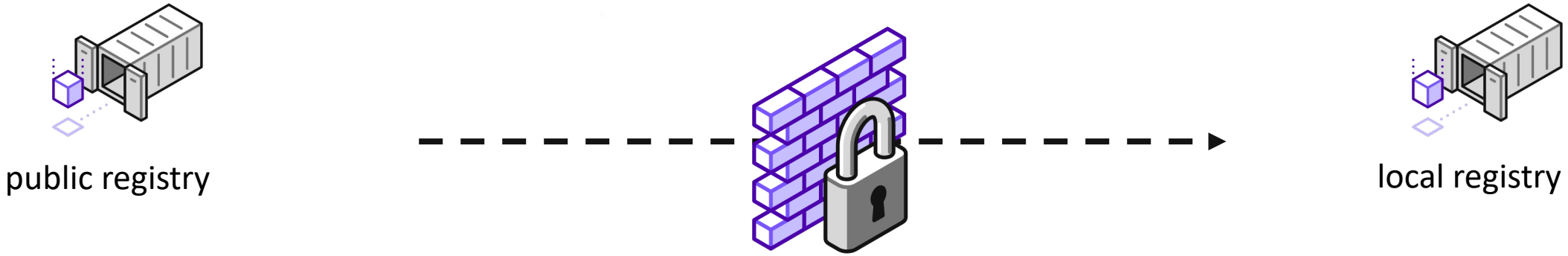
| Name | Status | Source registry | Replication Mode | Destination Registry:Namespace | Destination Repository Flattening | Trigger | Bandwidth | Description |
|---|---|---|---|---|---|---|---|---|
| registry.k8s.io | Enabled | registry.k8s.io | pull-based | Local : registry-mirror | Flatten 1 Level | Manual | Unlimited | - |
| cilium | Enabled | quay | pull-based | Local : capi-mirror | No Flatting | Manual | Unlimited | - |
| cilium_operator | Enabled | quay | pull-based | Local : capi-mirror | No Flatting | Manual | Unlimited | - |
| kube-vip | Enabled | ghcr.io | pull-based | Local : kube-vip | Flatten 1 Level | Manual | Unlimited | - |

Page size 5 1 - 4 of 4 items

# Local registry without Internet Access



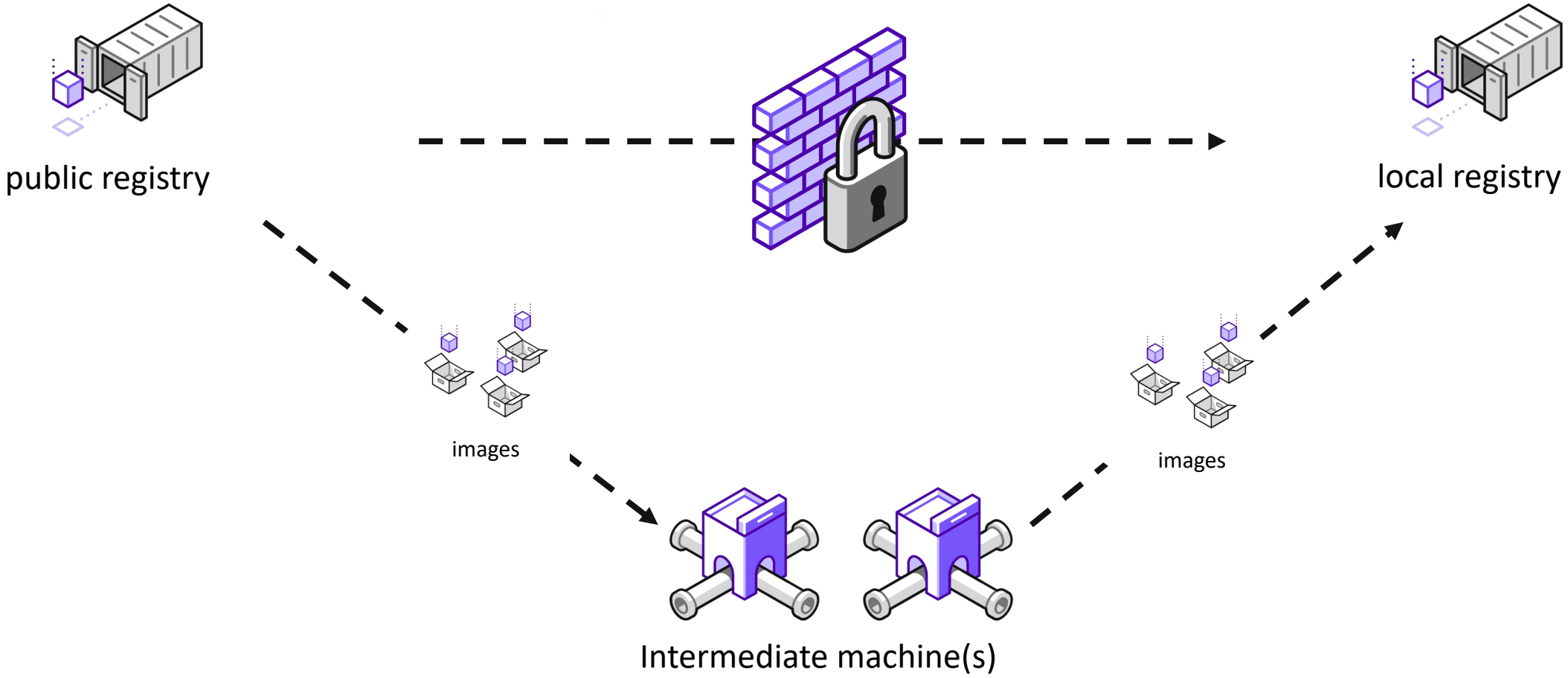
- Use intermediate machine(s) with dedicated tool



# Local registry without Internet Access



- Use intermediate machine(s) with dedicated tool

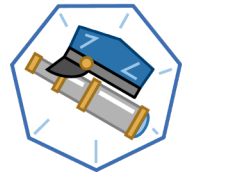


# Local registry without Internet Access

KubeCon CloudNativeCon Europe 2023

## skopeo

swiss army knife to migrate container images and image repositories



skopeo

```
# Install skopeo
~ ▶ (brew|dnf|yum|apt-get) install skopeo

# Copy each images to a local directory
~ ▶ skopeo sync -s docker -d dir <image> <localdir> --override-arch amd64 --override-os linux


# Copy directory content to a local registry
~ ▶ skopeo sync -s dir -d docker <localdir> <local-registry/project>
```

# Use your local registry: basic method



- Once your containers are in your local registry, just reference it in your manifest(s)

```
template:
  metadata:
    labels:
      app: my-nginx
  spec:
    containers:
    - image: nginx
      imagePullPolicy: Always
      name: web-static
      resources: {}
    dnsPolicy: ClusterFirst
    restartPolicy: Always
    schedulerName: default-scheduler
    securityContext: {}
    terminationGracePeriodSeconds: 30
```

```
template:
  metadata:
    labels:
      app: my-nginx
  spec:
    containers:
    - image: <local-registry/project>/nginx
      imagePullPolicy: Always
      name: web-static
      resources: {}
    dnsPolicy: ClusterFirst
    restartPolicy: Always
    schedulerName: default-scheduler
    securityContext: {}
    terminationGracePeriodSeconds: 30
```

# Use your local registry: helm method

KubeCon CloudNativeCon Europe 2023

Helm chart usually propose value to replace image path

```
# analyse Helm chart values to understood how to replace registry path
~ ▶ helm show values kyverno/kyverno

# Install the Kyverno Helm chart from a local registry
~ ▶ helm install kyverno kyverno/kyverno -n kyverno --create-namespace --set image.repository=<local-registry>/kyverno,initImage.repository=<local-registry>/kyvernopre,cleanupController.image.repository=<local-registry>/cleanup-controller
```

# Use your local registry: dynamic method



Use Kyverno to automatically change image path for pods

```yaml
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: replace-image-registry
  annotations:
    pod-policies.kyverno.io/autogen-controllers: none
spec:
  background: false
  rules:
    - name: replace-image-registry-pod-containers
      match:
        any:
        - resources:
            kinds: [ Pod ]
      mutate:
        foreach:
        - list: "request.object.spec.containers"
          patchStrategicMerge:
            spec:
              containers:
              - name: "{{ element.name }}"
                image: "{{ regex_replace_all_literal('^[^/]+', '{{element.image}}', '<local-registry>' )}}"
<... do the same for initContainers ...>
```

# Use your local registry: runtime method



You can configure containerd to override registry path

```
# Config containerd
~ ▶ cat /etc/containerd/config.toml | grep -B 1 config_path
[plugins."io.containerd.grpc.v1.cri".registry]
  config_path = "/etc/containerd/certs.d"

# Create config for each registry
~ ▶ find /etc/containerd/certs.d/
/etc/containerd/certs.d/
/etc/containerd/certs.d/_default
/etc/containerd/certs.d/_default/hosts.toml
/etc/containerd/certs.d/quay.io
/etc/containerd/certs.d/quay.io/hosts.toml

# Specify settings for each registry
~ ▶ cat /etc/containerd/certs.d/quay.io/hosts.toml
[host."https://<local-regsitry>/v2/quay.io"]
  capabilities = ["pull", "resolve"]
  override_path = true

~ ▶ cat /etc/containerd/certs.d/_default/hosts.toml
[host."https://<local-regsitry>/"]
  capabilities = ["pull", "resolve"]
```

# Enterprise life is different



## Limited Network

- Security rules
- Destination filtering
- Flow inspection
- Network Proxy
- User Authentication
- DMZ

**Common Security Posture**

## Isolated Network

- No internet access
- Bastion
- Recorded session

**High Security Posture**

## Constrained Network

- Limited bandwidth
- High latency
- Packet loss
- Traffic cost

**Edge scenario**

# Constrained Network



- Isolated network approach is also a good method for constrained network:
  - Local registry
  - Automatic image replication

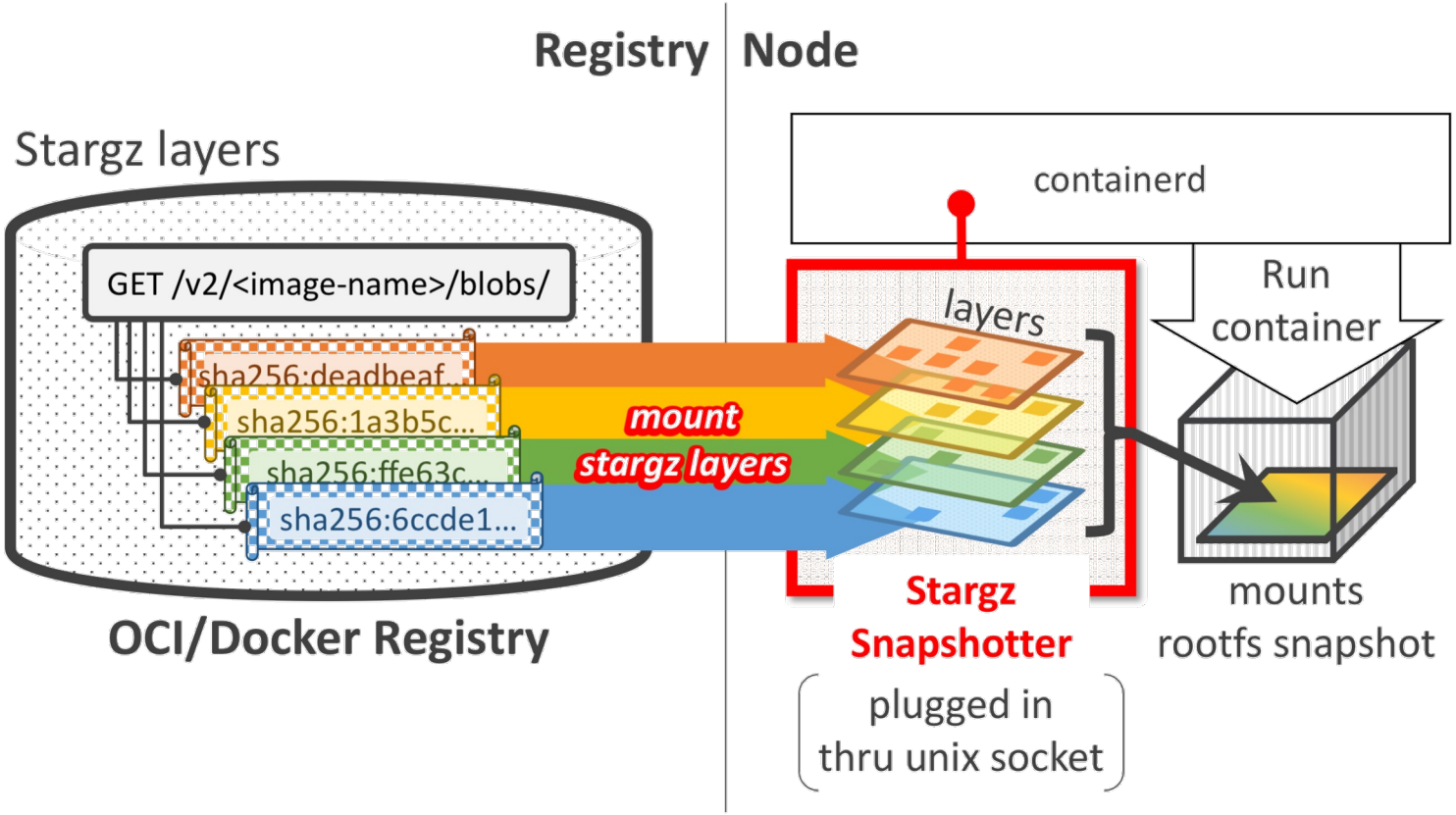
- Objective:
  - Always have the needed image onsite

- Drawback:
  - A new image pull always take time to be replicated
  - Delayed launch of new apps version

# Constrained Network: image streaming



- Lazy pulling with external snapshotter Stargz project
- Runtimes can start containers before the full image has been downloaded locally



# Constrained Network: image streaming



- Install stargz-snapshotter inside hosts
- Configure containerd to use it

```
# Enable stargz snapshotter for CRI
[plugins."io.containerd.grpc.v1.cri".containerd]
  snapshotter = "stargz"
  disable_snapshot_annotations = false

# Plug stargz snapshotter into containerd
[proxy_plugins]
  [proxy_plugins.stargz]
    type = "snapshot"
    address = "/run/containerd-stargz-grpc/containerd-stargz-grpc.sock"
```

- Rebuild image or optimize existing (can be managed at the registry level)

# Constrained Network: image streaming

KubeCon CloudNativeCon Europe 2023

```
# Launch a pod using a non optimized image
~ ▶ kubectl create deployment my-nginx --image=nxlab.fr/cache/web-static:latest
deployment.apps/my-nginx created
~ ▶ kubectl get event | grep Pulled
....  Successfully pulled image "nxlab.fr/cache/web-static:latest" in 3m59.589685046s


# Launch a pod using a eStargz optimized image
~ ▶ kubectl create deployment my-nginx --image=nxlab.fr/cache/web-static:latest-esgz
deployment.apps/my-nginx created
~ ▶ kubectl get event | grep Pulled
....  Successfully pulled image "nxlab.fr/cache/web-static:latest-esgz" in 2.603734251s
```

# Wrap-up: K8s can survive with



## Limited Network

- Open correct network flows
- Proxy config
- Pod auto config with Kyverno

## Isolated Network

- Local registry
- Registry dynamic replication / two steps copy
- Containers image path dynamic config (Kyverno/Containerd)

## Constrained Network

- Isolated Network architecture
- Containers streaming (eStargz/Containerd)

# Resources



- Kyverno: https://kyverno.io
- Harbor: https://goharbor.io
- Skopeo: https://github.com/containers/skopeo
- Containerd image mirror docs: https://github.com/containerd/containerd/blob/main/docs/hosts.md
- Stargz-snapshotter: https://github.com/containerd/stargz-snapshotter
- Harbor Acceleration service: https://github.com/goharbor/acceleration-service

Please scan the QR Code above
to get content and leave feedback on this session


# Thank you!

@tuxtof on

Christophe Jauffret on

tuxtof@funkolab.net

## visit us on booth S30