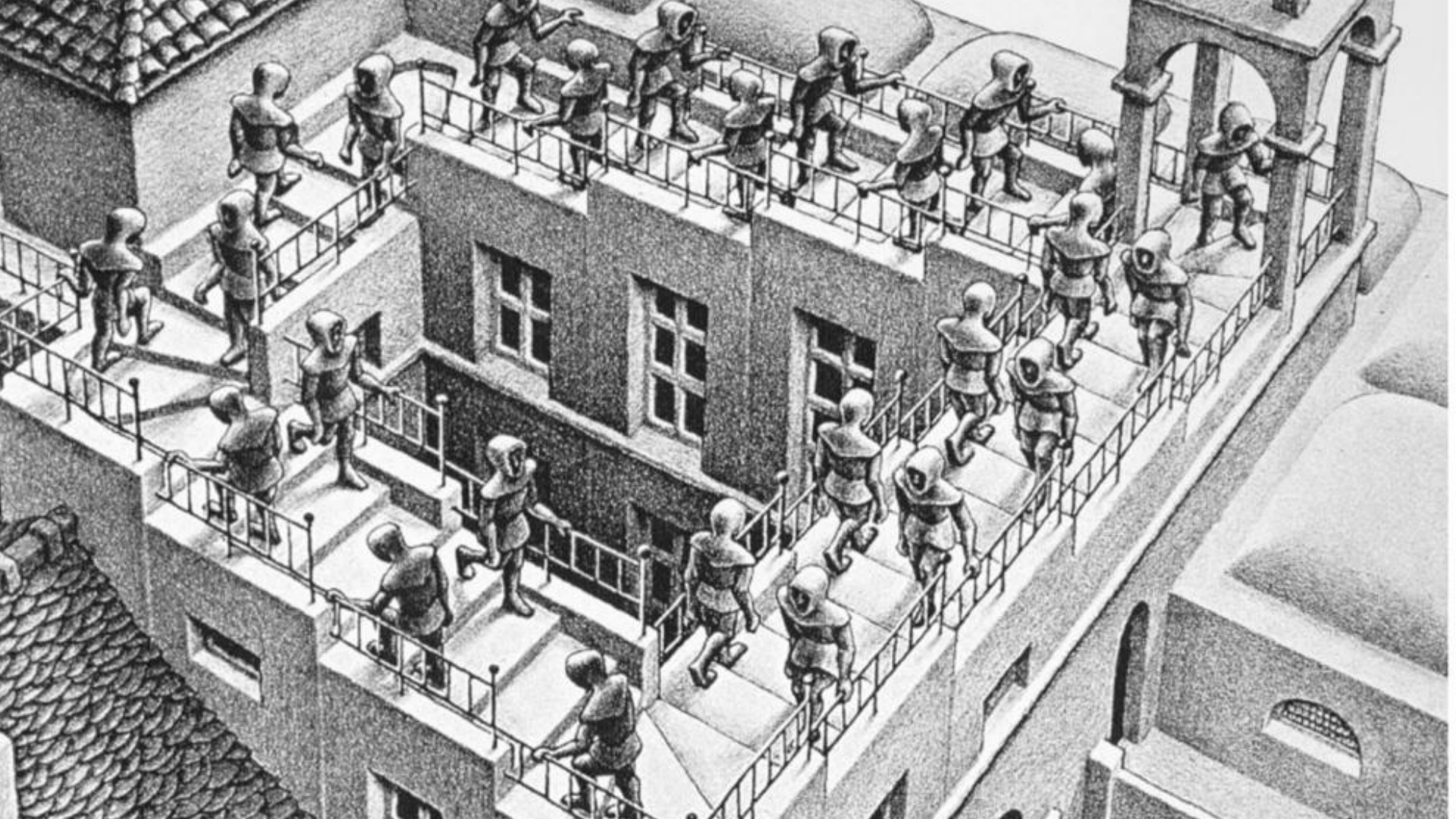
Operating kube-apiserver

Without Hiccups

Stefan Schimanski & David Eads



Orchestrate

Config changes
Upgrades
Cert Rotation

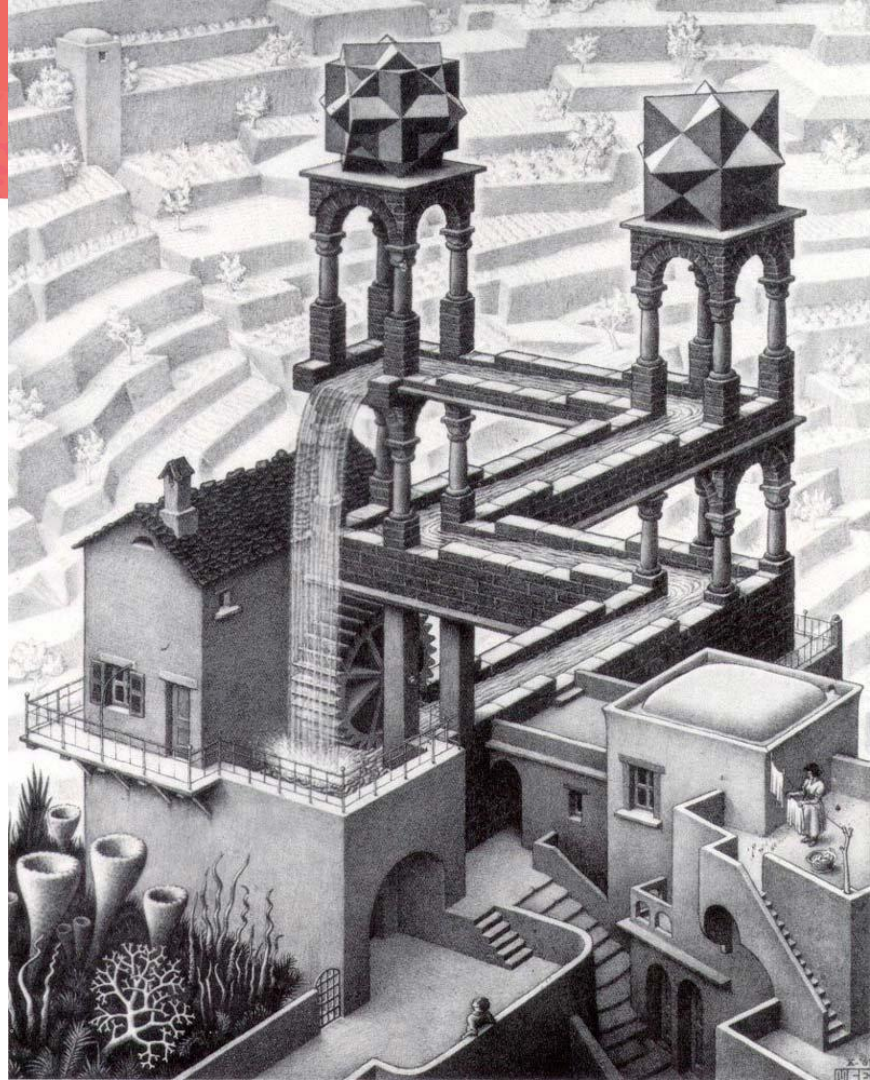
} without hiccups

bootkube, the vision

- **initialize** a cluster **with static pods**
- use them to **create DaemonSets**
- **kill static pods**

- **all future updates** using **DaemonSet**

- but what if we **crash**...



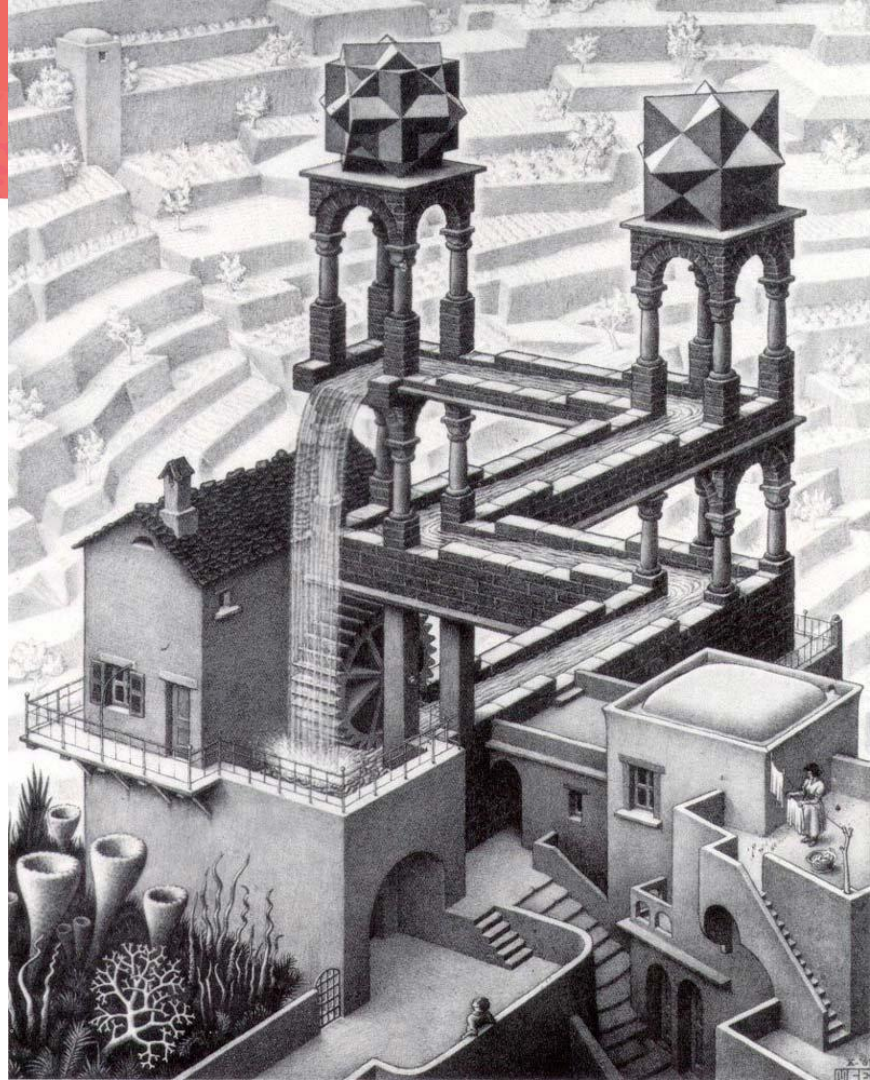
bootkube, the vision

- **initialize** a cluster **with static pods**
- use them to **create DaemonSets**
- **kill static pods**

- **all future updates** using **DaemonSet**

- but what if we **crash**…

We have the **static pod checkpointer**!



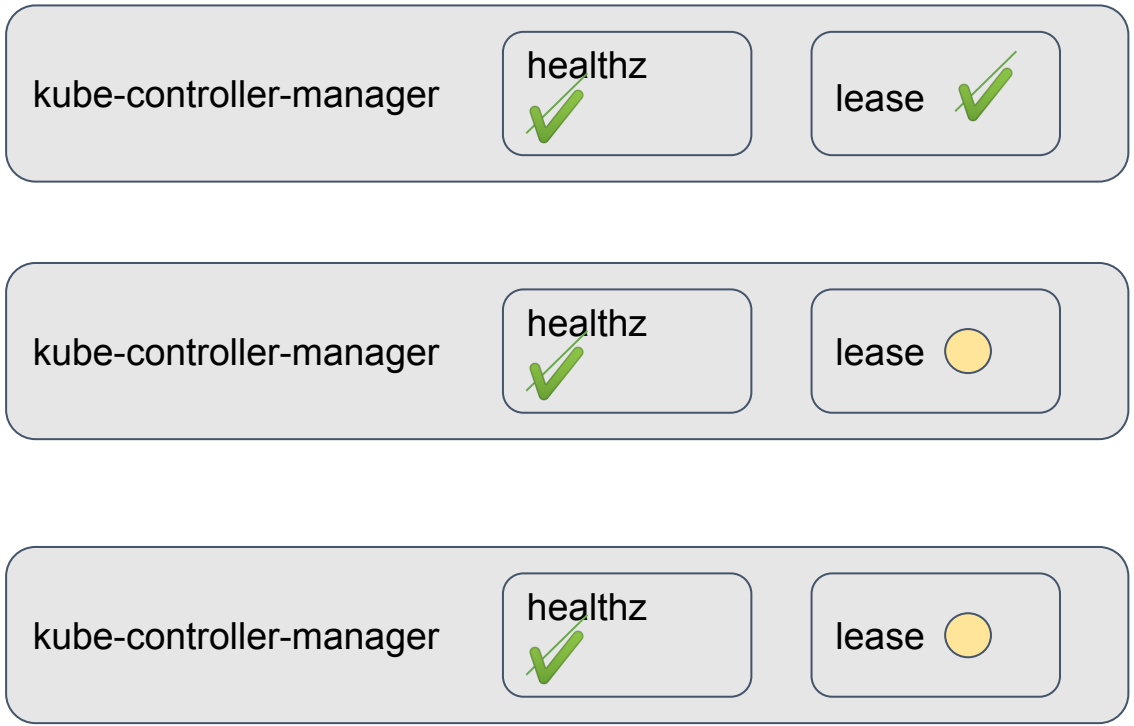
bootkube checkpointer / recover

- What does the **checkpointer** actually do?

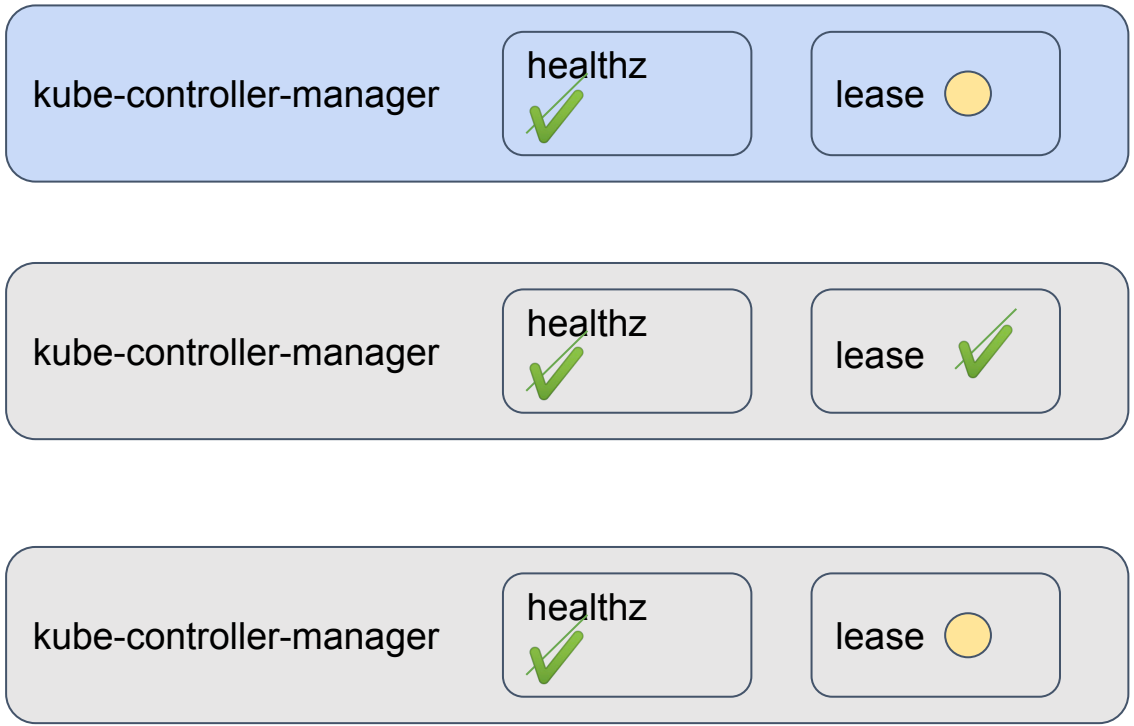
  tl/dr:
  copying pod manifests, secrets, configmaps to the filesystem
  and **rewriting** the **pod yaml** … to hopefully **run as static pod**.

- **Does it work?** Most of the time, but happens if ...

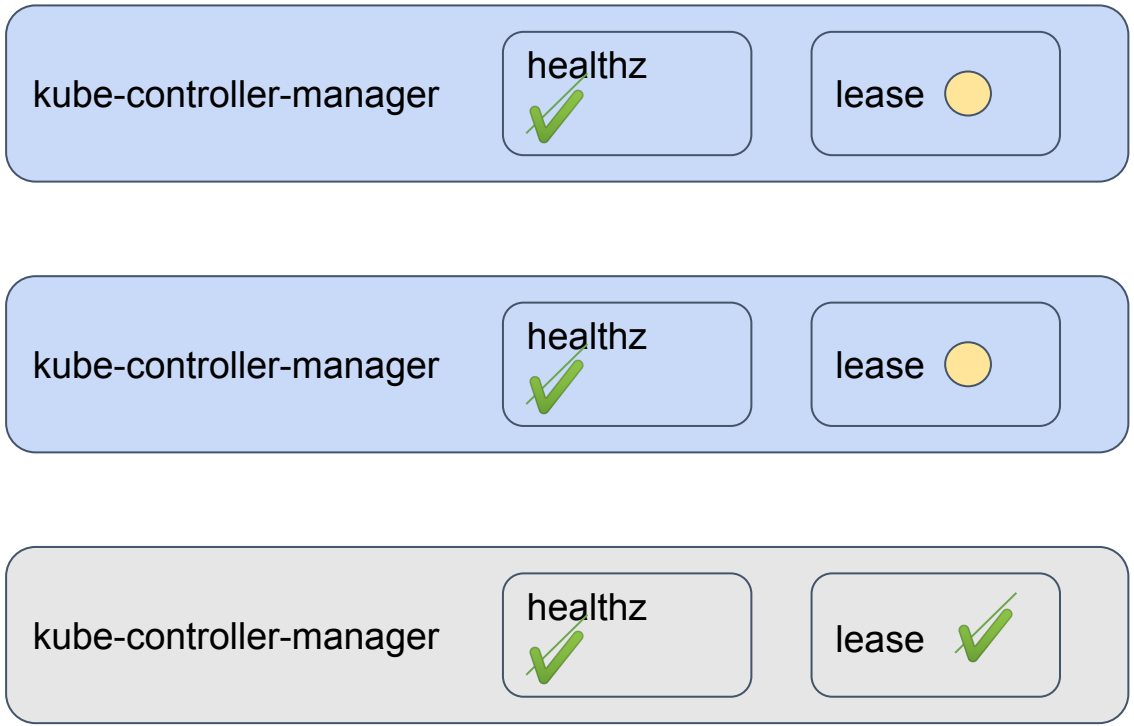
bootkube: update of doom



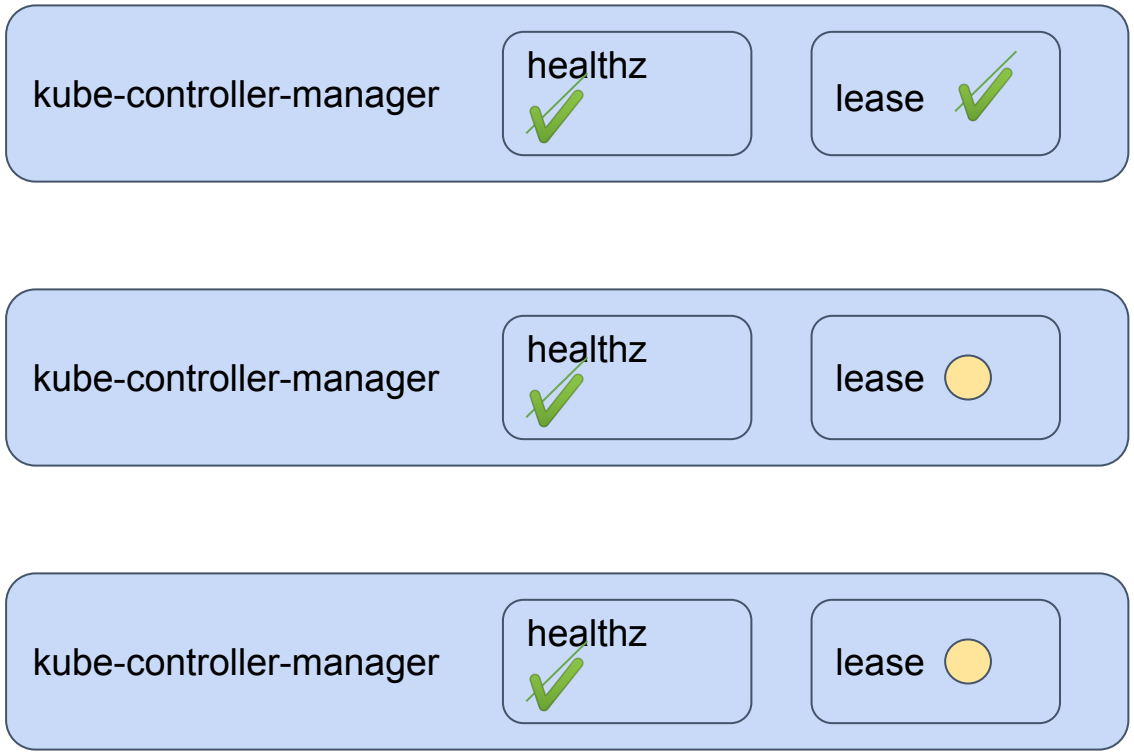
bootkube: update of doom



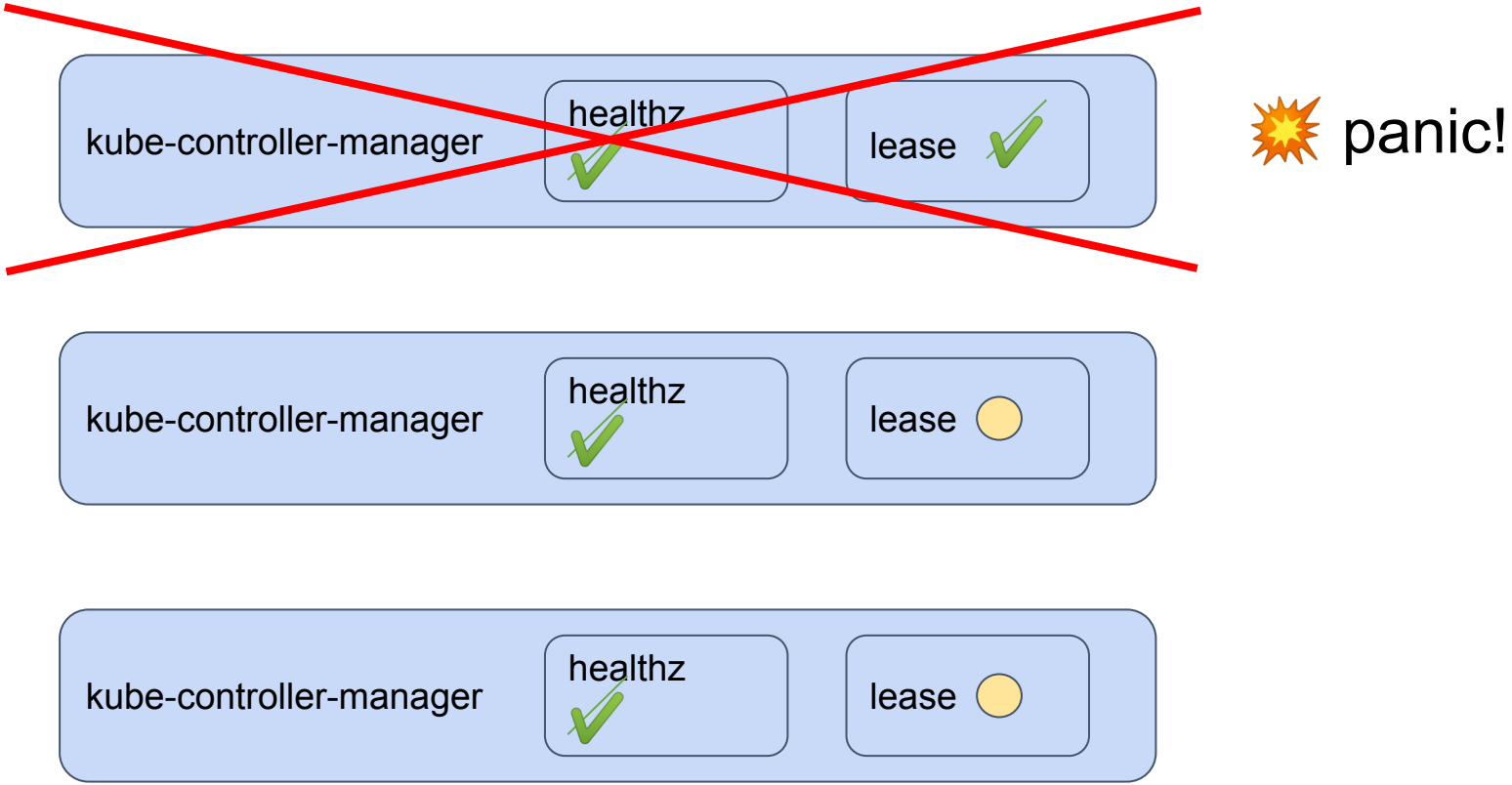
bootkube: update of doom



bootkube: update of doom

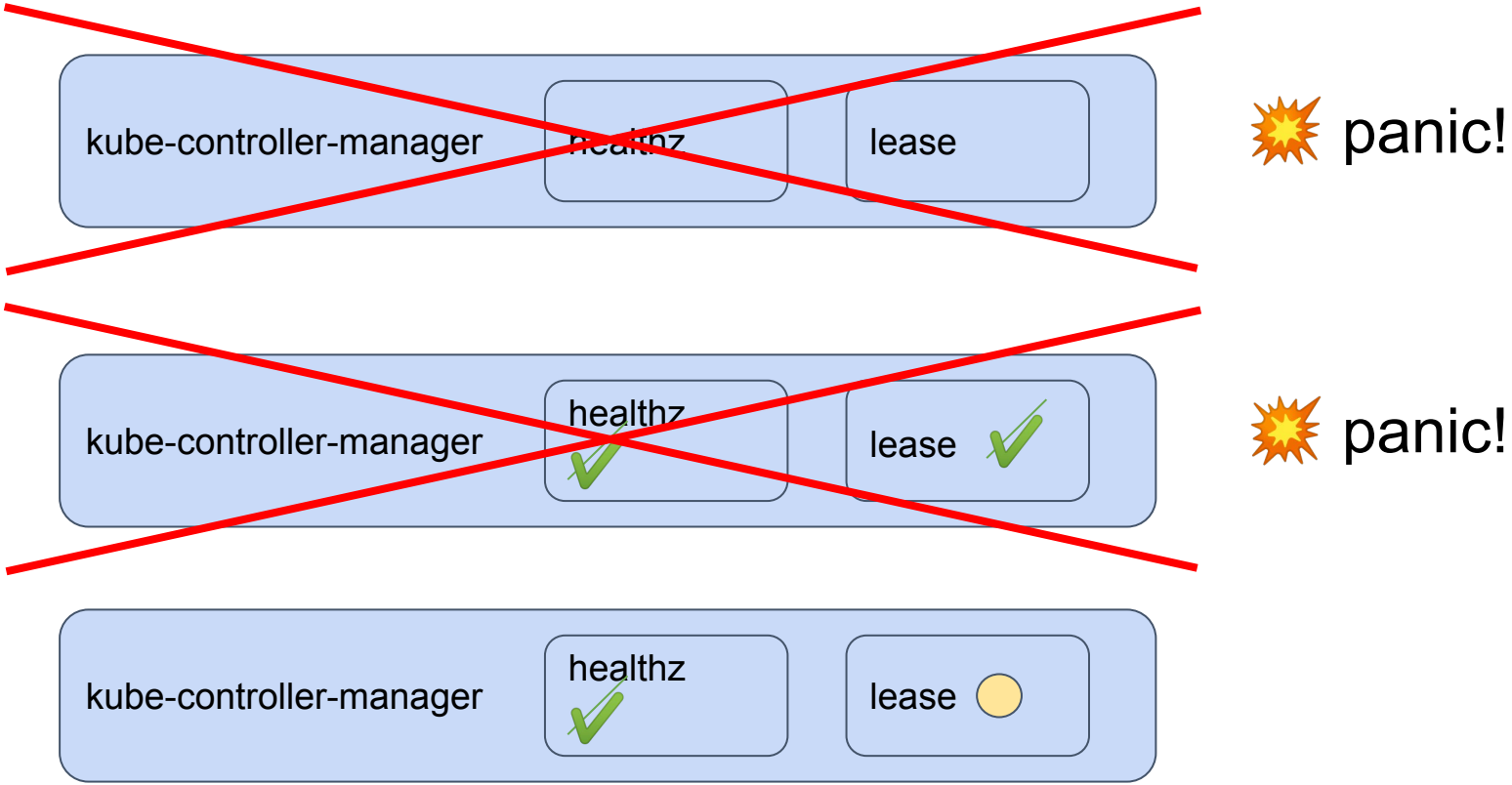


bootkube: update of doom

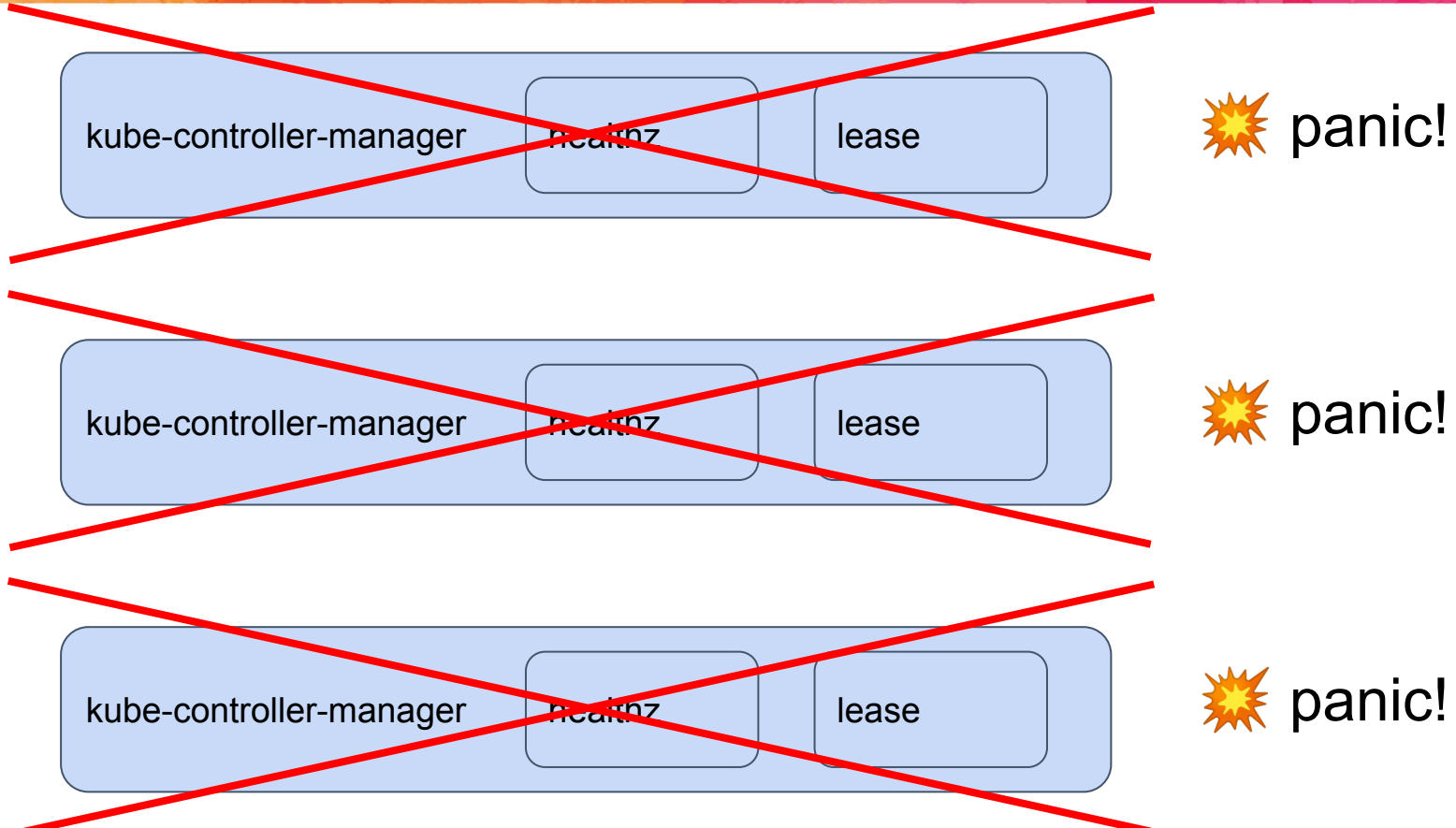


panic!

bootkube: update of doom



bootkube: update of doom



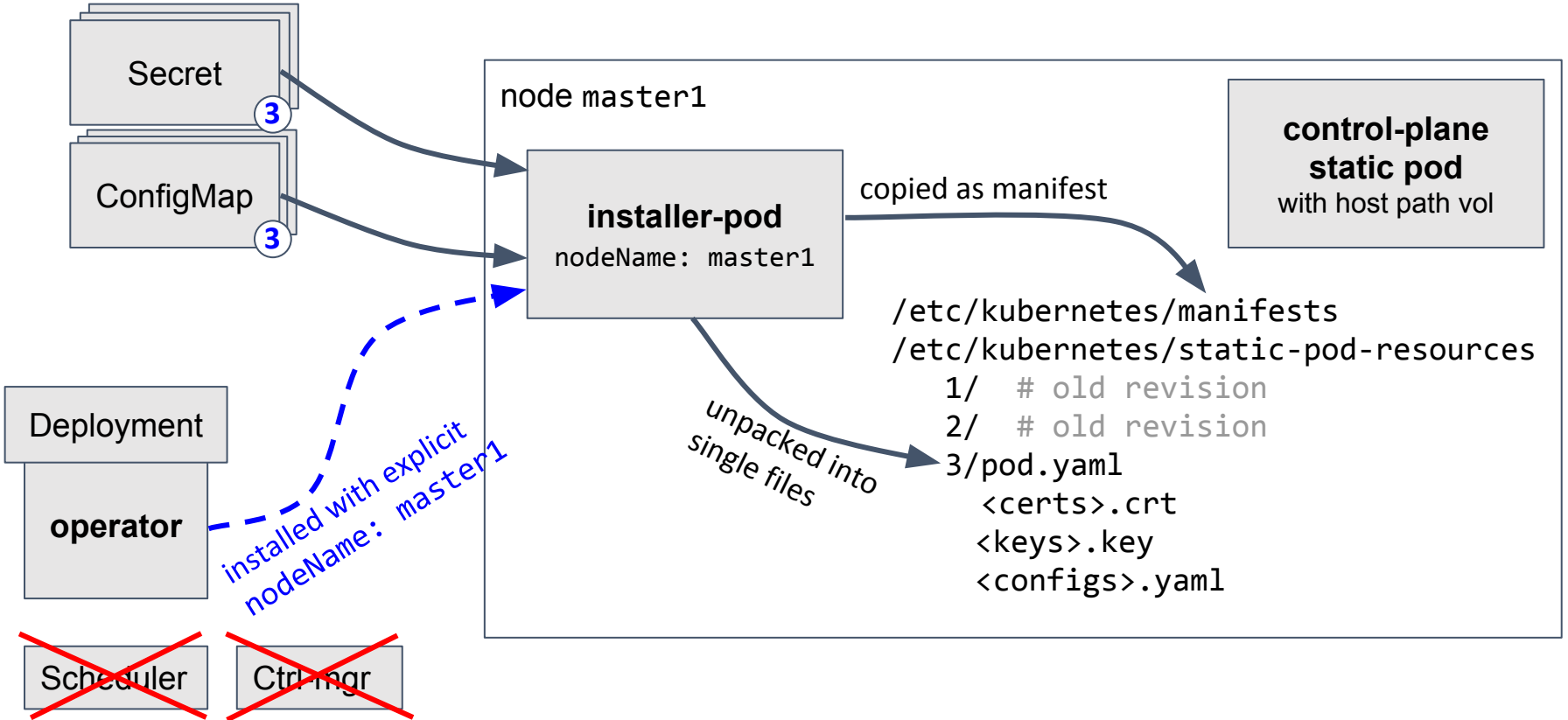
the alternative: static pods

1. Avoid pivot
2. Avoid checkpointer
3. Always run the same pod
4. Always tolerate kubelet connection failures
5. Always have a local backup
6. No cyclical dependencies

Self-hosting is awesome, but in a down-to-earth way:

control-plane as static pods – operator as DaemonSet

static pod management



static pod management

1. Create a **set of immutable configmaps & secrets**: a **revision**.
2. Create a **installer-pods**, forced to a particular master without scheduler

   `spec.NodeName: master<n>`

   with **hosts mounts**: **static pod manifest** & **static resources directories**
3. The **installer-pod copies**: **configmaps** & **secrets** → **static resources** dir
   static pod manifest → **static pod** dir
4. **Wait for new static pod** to become ready
5. Then **move to next node**
6. If you hit an “update of doom”, you can retry by creating new revisions and static pods because no workload resource is required.

without hiccups

- make sure **clients don't get immediately dropped**
- make sure you're **not-ready**, **but** you are **healthy**
- make sure that your **load-balancer REALLY stops sending traffic**
- make sure that the **service network stops sending traffic**

Errors everywhere

Failed to list *core.Service: Get https://172.30.0.1/api/v1/services?limit=500&resourceVersion=0:dial tcp 172.30.0.1:6443: **connect: connection refused**

I0326 20:03:52.589926 3853 streamwatcher.go:107] Unable to decode an event from the watch stream: http2: **server sent GOAWAY and closed the connection**; LastStreamID=53, ErrCode=NO_ERROR, debug=""

F1030 18:27:51.842709 4254 server.go:262] cannot create certificate signing request: Post https://1.2.3.4:6443/apis/certificates.k8s.io/v1beta1/certificatesigningrequests: **EOF**

I0417 12:18:54.309074 1 streamwatcher.go:103] Unexpected EOF during watch stream event decoding: **unexpected EOF**

apiservice/v1.apps.openshift.io: not available: no response from https://172.30.83.61:443: Get https://172.30.83.61:443: dial tcp 172.30.83.61:443: **connect: no route to host**.

no HTTP server listening

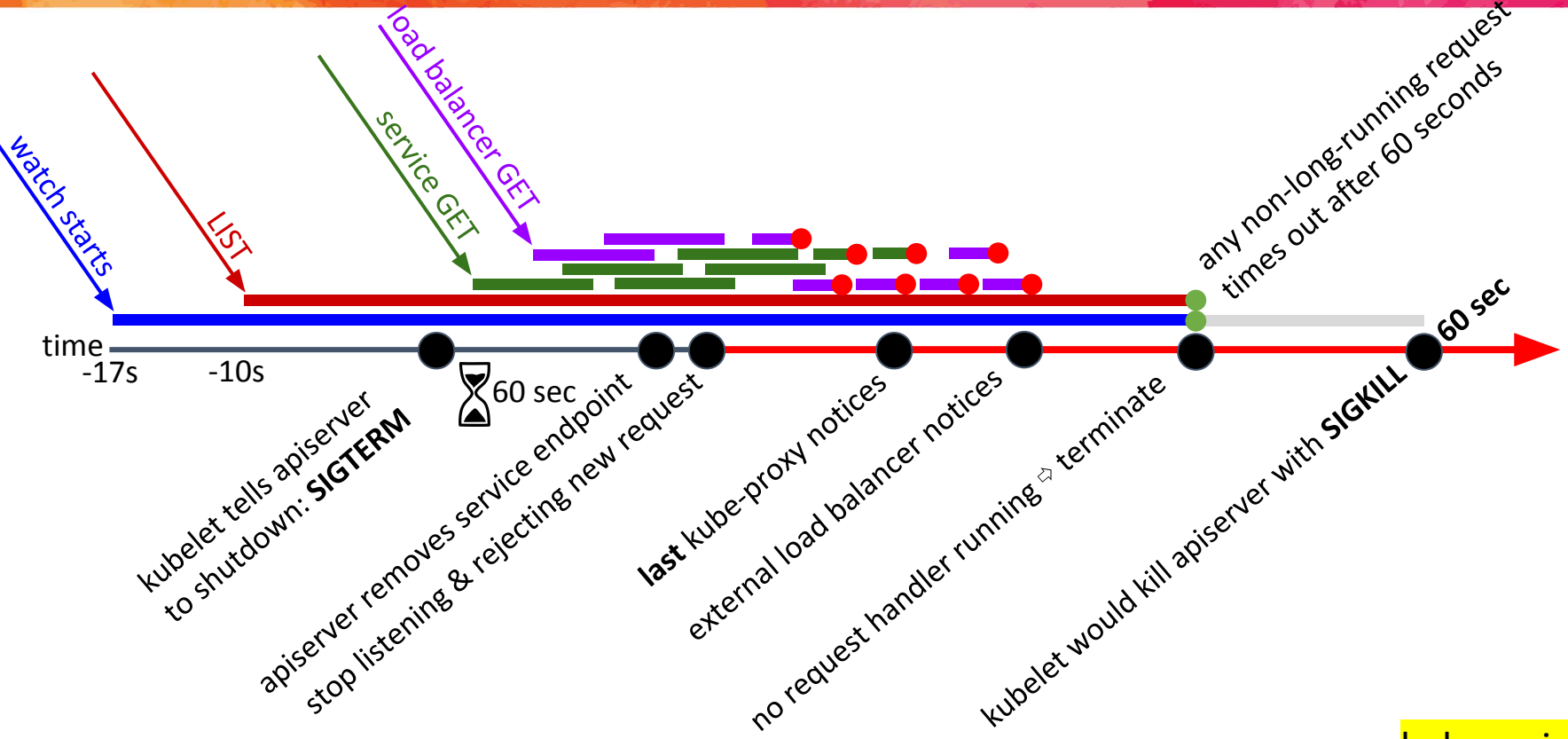
long-running request HTTP/2

cut-off request non-long-running

long-running request

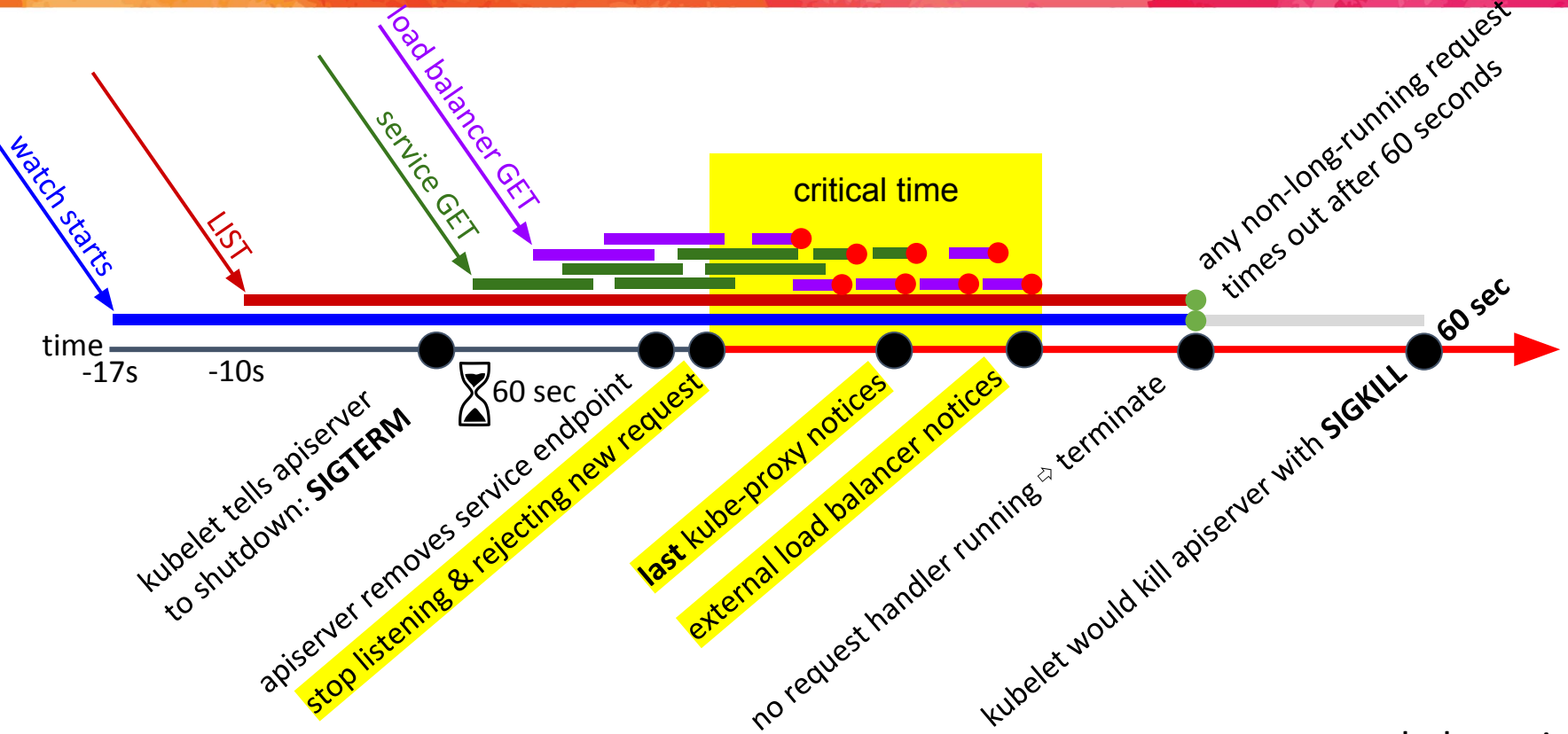
probably network issue or rebooting node

Graceful Termination



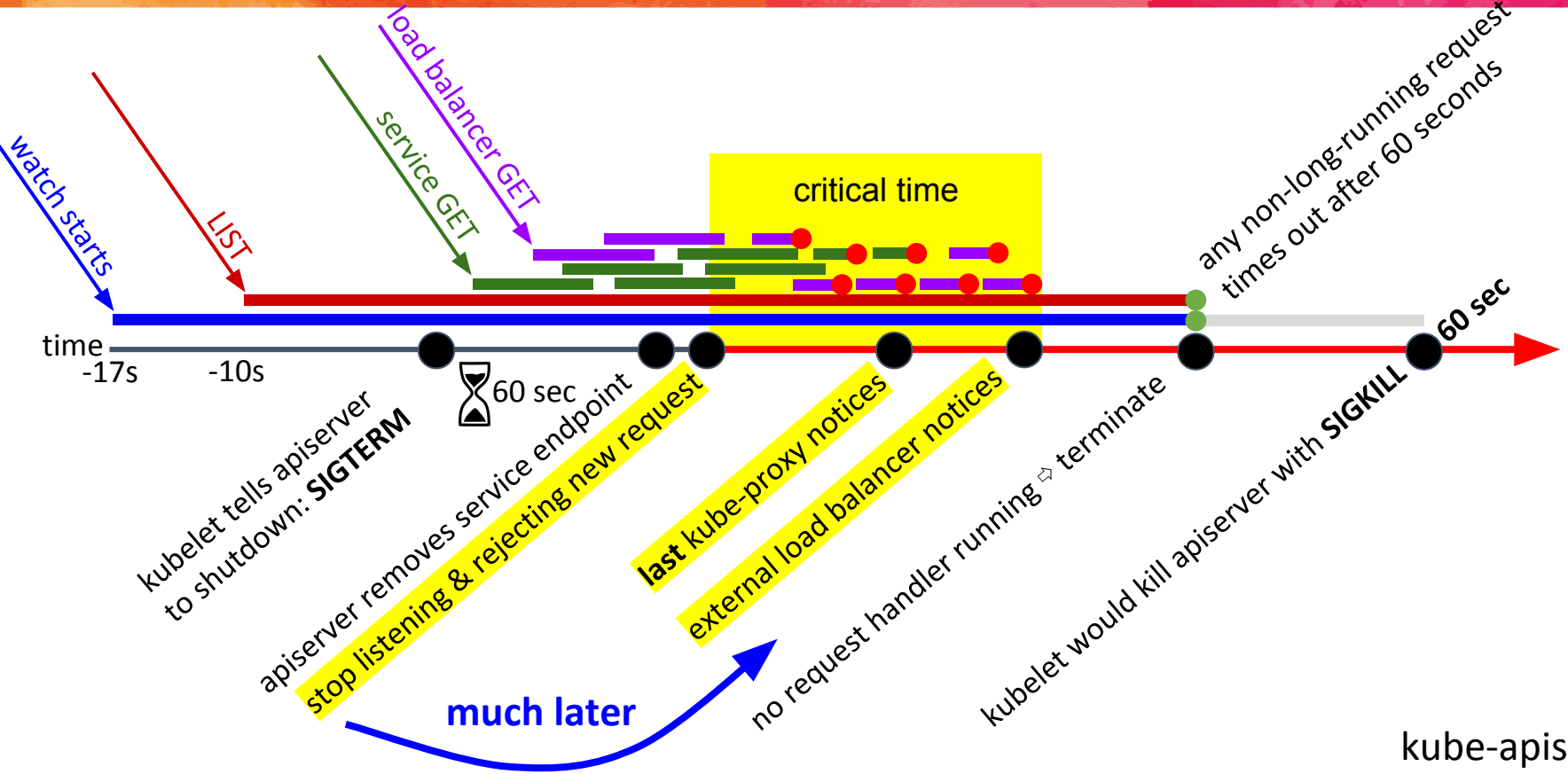
kube-apiserver

Graceful Termination

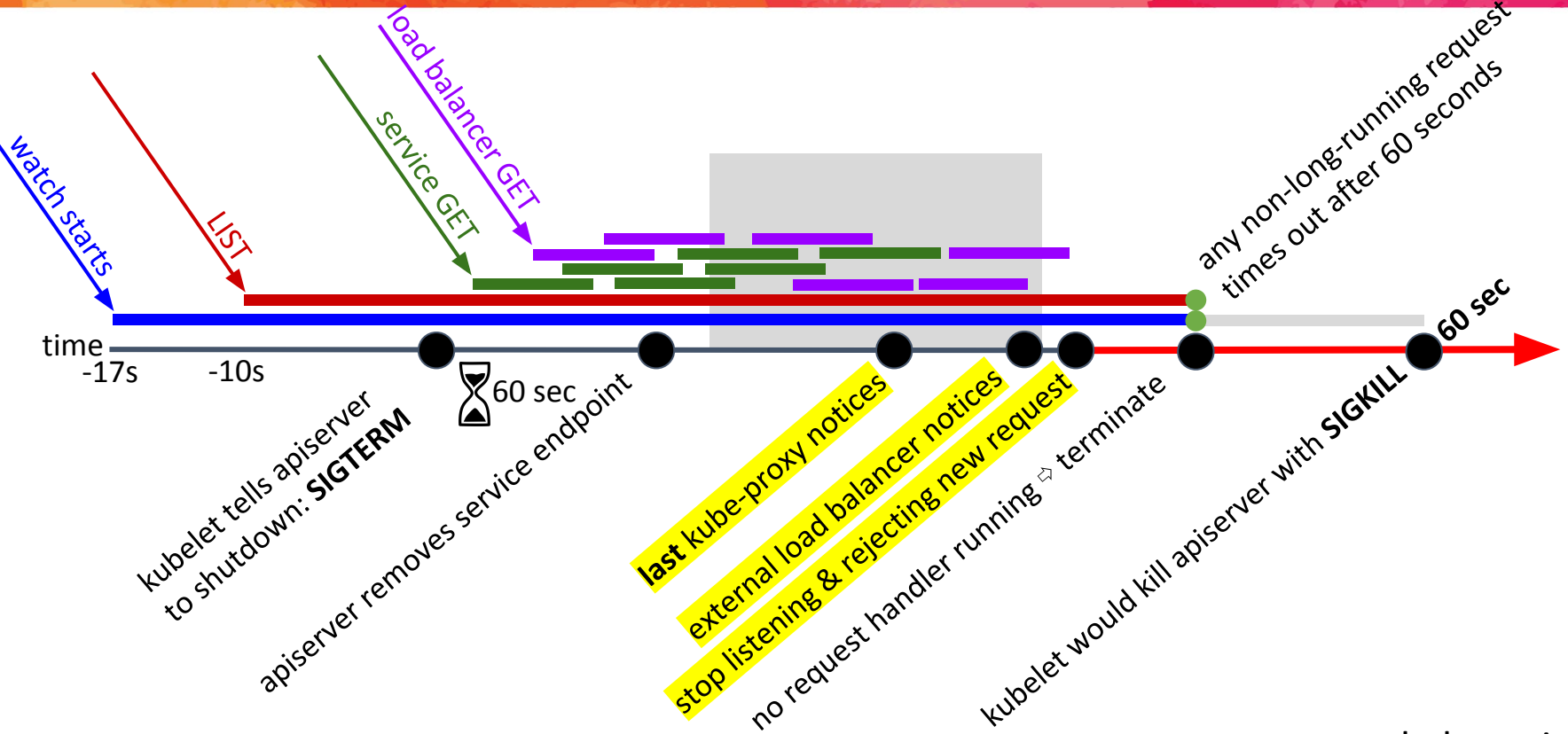


kube-apiserver

Graceful Termination



Graceful Termination



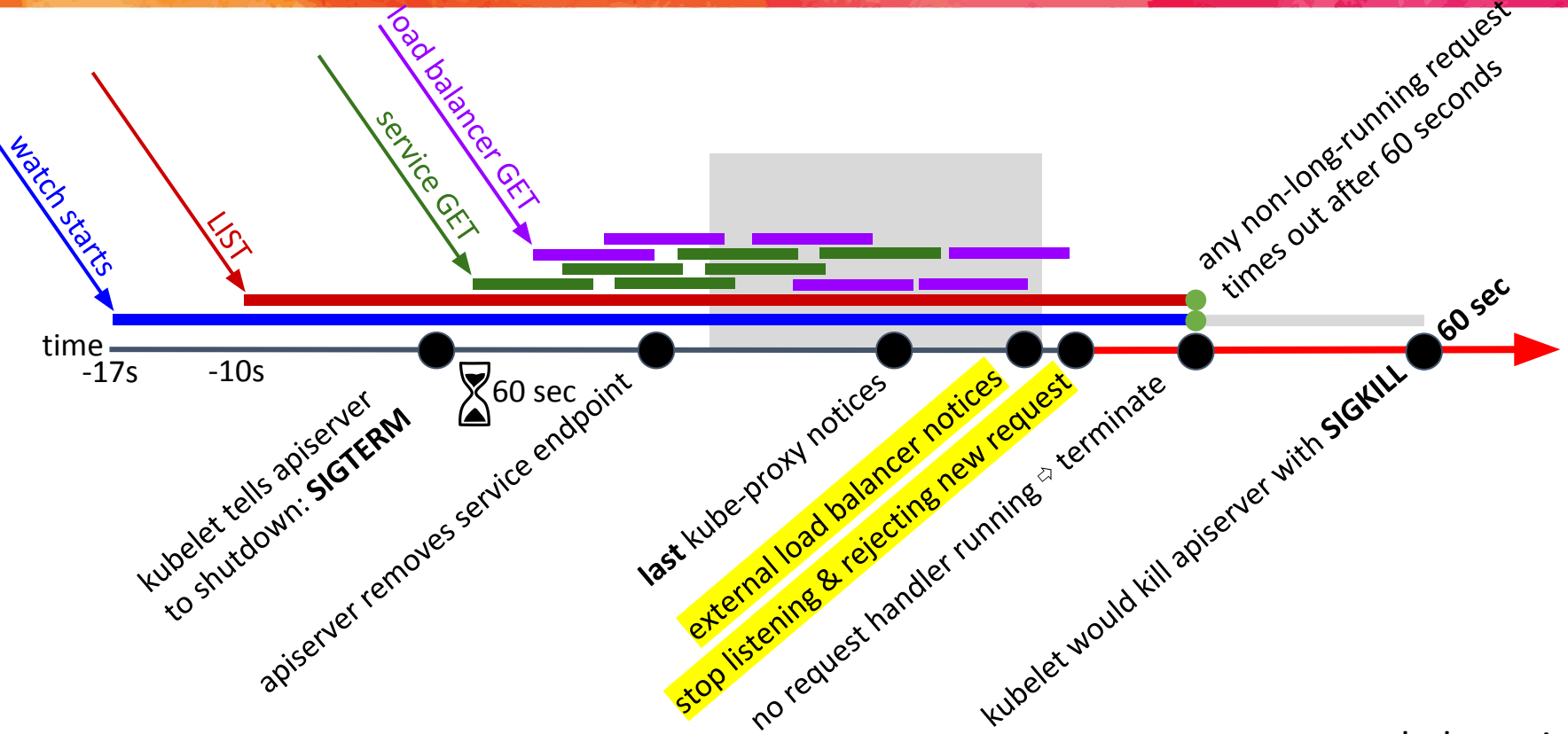
kube-apiserver

Minimum Shutdown Duration

- **immediately** “tell” internal & external **load balancers**
- **keep listening & responding** normally for **n seconds**
- give the cluster **time to adapt**
- **then stop listening** and do graceful termination.

does not exist today
PR 74416

Graceful Termination



kube-apiserver

Liveness vs. Readiness – healthz or readyz

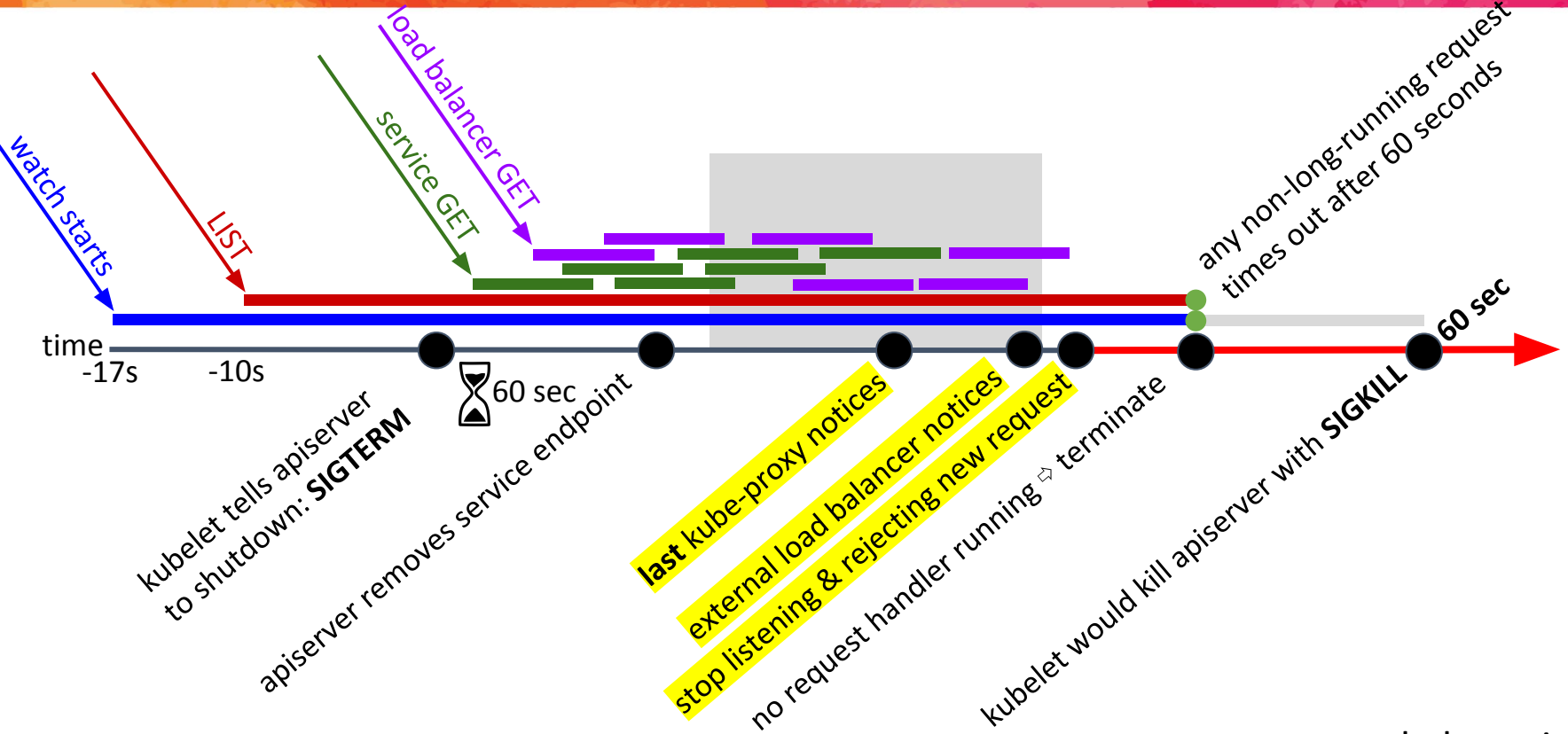
- **/healthz** error => kubelet kills the pod
- **/readyz** error => endpoint controller removes endpoint **internal**
  load balancers remove target **external**

We need:

1. **SIGTERM** to kube-apiserver
   a. **/readyz error immediately**
   b. **remove endpoint** from `default/kubernetes`
2. wait **minimum-shutdown-period**
3. **graceful shutdown** up to 60 sec

/readyz does not exist today

Graceful Termination



kube-apiserver

Why we do all this

Config changes

Upgrades

Cert Rotation

} without hiccups

certificate settings today

--client-ca-file	CA bundle used to verify client certificate connections from clients and identify users. (I am Bob). Must be able to verify `kube-controller-manager --cluster-signing-cert-file` or `kubelet --rotate-certificates` will fail.
--requestheader-client-ca-file	CA bundle used to verify client certificate connections from front proxies that are asserting the identity of user. (This request is from Bob). Must be able to verify `kube-apiserver --proxy-client-cert-file` or aggregation in the cluster will fail by default.
--kubelet-certificate-authority	CA bundle used to verify kubelets for connections from KAS to kubelet. (Think logs,exec,etc). Must be able to verify `kubelet --tls-cert-file`. Must be able to verify `kube-controller-manager --cluster-signing-cert-file` or `kubelet --rotate-server-certificates` will fail.
--kubelet-client-certificate	Client cert used to identify KAS to the kubelets. Must be verifiable by `kubelet --client-ca-file`.
--kubelet-client-key	Client key used to identify KAS to the kubelets
--proxy-client-cert-file	Client cert used to identify KAS to aggregated API servers as a front proxy. Must be verifiable by `kube-apiserver --requestheader-client-ca-file` or aggregation in the cluster will fail by default
--proxy-client-key-file	Client key used to identify KAS to aggregated API servers as a front proxy
--service-account-key-file	RSA keys used to verify ServiceAccount tokens. Must be able to verify `kube-controller-manager --service-account-private-key-file` for all keys you want to continue working.
kube-controller-manager	**What's it for**
--client-ca-file	CA bundle used to verify client certificate connections from clients and identify users. (I am Bob)
--tls-cert-file	Serving cert used to serve requests
--tls-private-key-file	Serving key used to serve requests

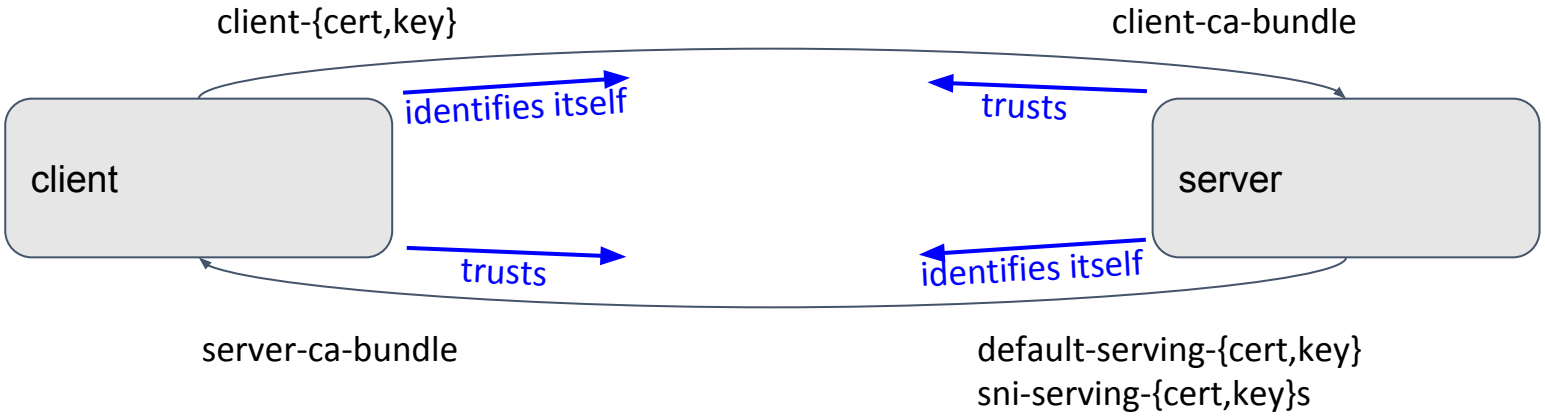
certificate settings today

--tls-private-key-file	Serving key used to serve requests not matching SNI
--tls-sni-cert-key	Special flag format to specify hostname-pattern,cert,key tuples to serve matching SNI requests. If used for kubernetes.default.service, must be verifiable with `kube-controller-manager --root-ca-file`.
--client-ca-file	CA bundle used to verify client certificate connections from clients and identify users. (I am Bob). Must be able to verify `kube-controller-manager --cluster-signing-cert-file` or `kubelet --rotate-certificates` will fail.
--requestheader-client-ca-file	CA bundle used to verify client certificate connections from front proxies that are asserting the identity of user. (This request is from Bob). Must be able to verify `kube-apiserver --proxy-client-cert-file` or aggregation in the cluster will fail by default.
--kubelet-certificate-authority	CA bundle used to verify kubelets for connections from KAS to kubelet. (Think logs,exec,etc). Must be able to verify `kubelet --tls-cert-file`. Must be able to verify `kube-controller-manager --cluster-signing-cert-file` or `kubelet --rotate-server-certificates` will fail.
--kubelet-client-certificate	Client cert used to identify KAS to the kubelets. Must be verifiable by `kubelet --client-ca-file`.
--kubelet-client-key	Client key used to identify KAS to the kubelets
--proxy-client-cert-file	Client cert used to identify KAS to aggregated API servers as a front proxy. Must be verifiable by `kube-apiserver --requestheader-client-ca-file` or aggregation in the cluster will fail by default
--proxy-client-key-file	Client key used to identify KAS to aggregated API servers as a front proxy
--service-account-key-file	RSA keys used to verify ServiceAccount tokens. Must be able to verify `kube-controller-manager --service-account-private-key-file` for all keys you want to continue working.
kube-controller-manager	**What's it for**
--client-ca-file	CA bundle used to verify client certificate connections from clients and identify users. (I am Bob)
--tls-cert-file	Serving cert used to serve requests
--tls-private-key-file	Serving key used to serve requests
--cluster-signing-cert-file	Signing cert used to issue approved CSR requests. Must be verifiable with `kube-apiserver --kubelet-client-certificate` and `kube-apiserver --client-ca-file` or `kubelet --rotate-certificates` will fail.
--cluster-signing-key-file	Signing key used to issue approved CSR requests
--requestheader-client-ca-file	CA bundle used to verify client certificate connections from front proxies that are asserting the identity of user. (This request is from Bob)
--root-ca-file	CA bundle injected into ServiceAccount token secrets. It is **only** intended to be used to verify a connection to the kube-apiserver on the service network. All other uses are either wrong or coincidence. Must be able to verify `kube-apiserver --tls-cert-file`
--service-account-private-key-file	RSA key used to sign ServiceAccount tokens. Must be verifiable by `kube-apiserver --service-account-key-file` or ServiceAccounts will not be able to

certificate settings today

KubeCon CloudNativeCon Europe 2019

certs: super basic mTLS

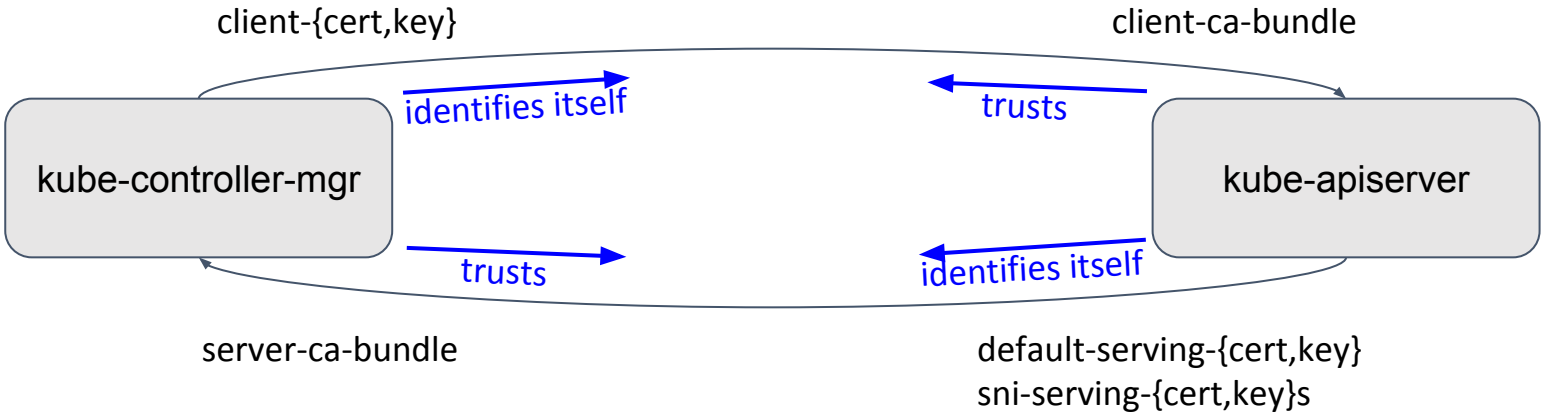


possible lag (minutes!) **in both directions** => **keep** old **CAs around**

lack of synchronization => **be slow**

unbounded lag: nodes are down or **golden images**

certs: super basic mTLS



possible lag (minutes!) **in both directions** => **keep** old **CAs around**

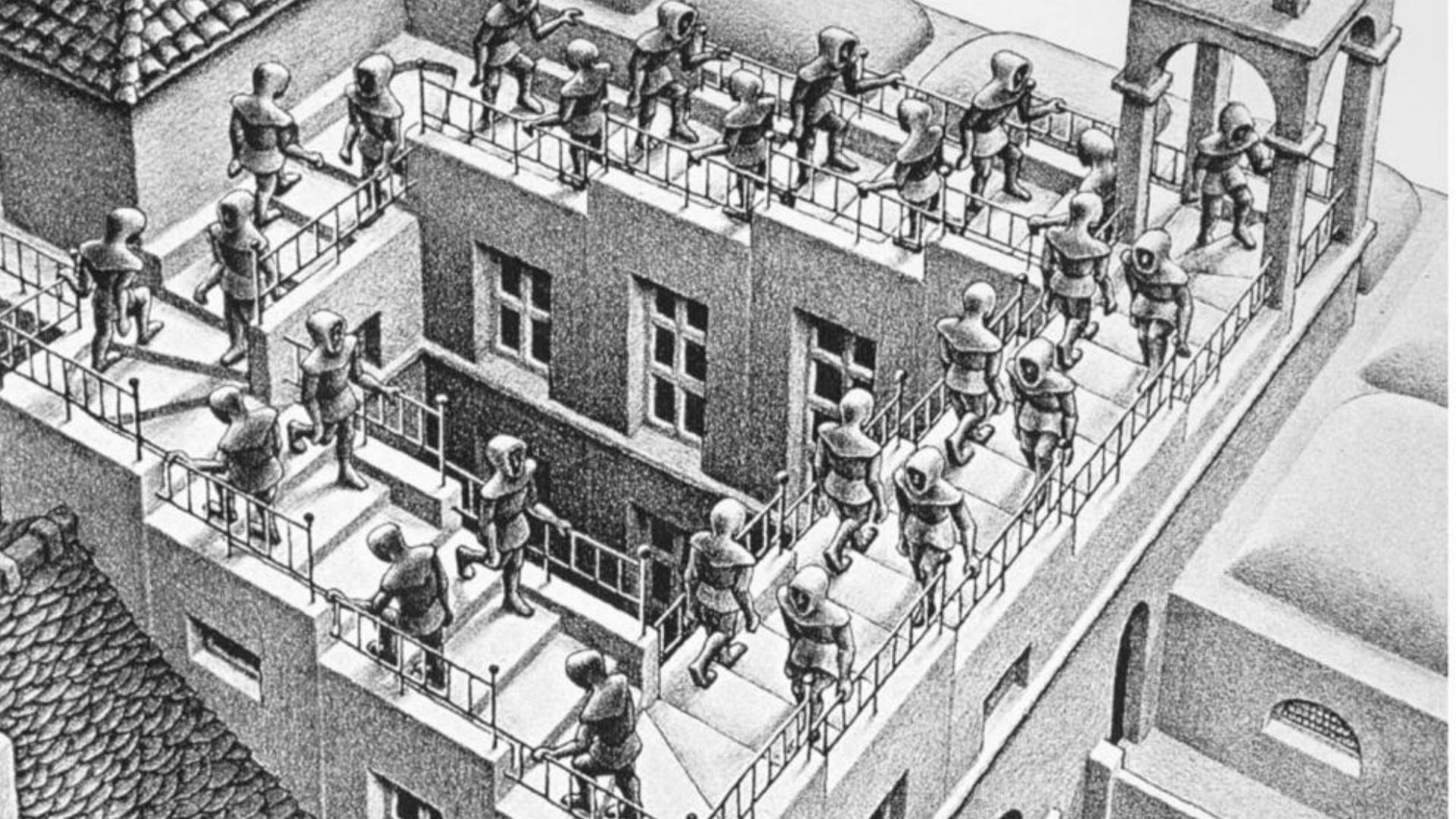
lack of synchronization => **be slow**

unbounded lag: nodes are down or **golden images**

certs: manage them automatically

- Relationships are hard
  - kubelet
    - **--rotate-certificates**
    - **--rotate-server-certificates**
  - kube-controller-manager
    - **--cluster-signing-cert-file**
    - **--cluster-signing-key-file**
  - kube-apiserver
    - **--client-ca-file**
    - **--kubelet-certificate-authority**

- Cluster-admins only care about: **kube-apiserver serving cert/key**



Orchestrate

Config changes
Upgrades
Cert Rotation

} without hiccups