KubeCon | CloudNativeCon

Europe 2018

What does "production ready" really mean for a Kubernetes cluster?

Lucas Käldström

4th of May 2018 - KubeCon Copenhagen

$ whoami

CLOUD NATIVE COMPUTING FOUNDATION AMBASSADOR

Lucas Käldström, Upper Secondary School Student, just turned 18

CNCF Ambassador, **Certified Kubernetes Administrator** and **Kubernetes SIG Lead**

CERTIFIED kubernetes ADMINISTRATOR

Speaker at **KubeCon** in **Berlin & Austin** in 2017

Kubernetes approver and subproject owner, active in the community for ~3 years

Driving **luxas labs** which currently performs contracting for Weaveworks

A guy that has never attended a computing class

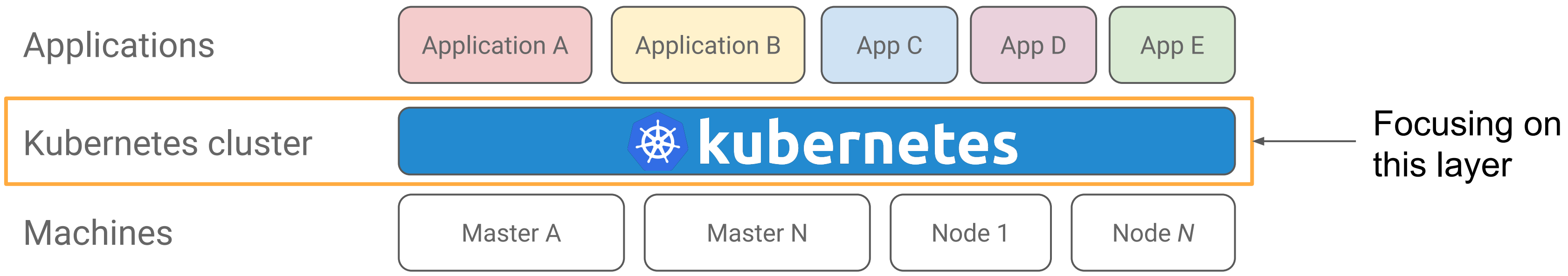
Agenda

1. Define the buzzwords!
   a. What does “production-ready” mean to you?
   b. What are the requirements for a highly available cluster?
2. What to think about when securing the cluster
   a. TLS certificates for all components
   b. Enable and set up RBAC (Role Based Access Control)
   c. Attack vectors you might not have thought about before

Agenda

3. Make the cluster highly-available if needed
    a. Do you need it?
    b. How to set up a HA cluster with kubeadm
    c. “Attack vectors” you might not have thought about before
4. Use the *Cluster API* for controlling the cluster declaratively
    a. Intro to the Cluster API
    b. How to set up Kubernetes using the Cluster API and upgrade/rollback

Which layer are you talking about?



KubeCon | CloudNativeCon

Europe 2018

I. Define what “production-ready” means to you

Buzzwords all around...

“The cluster is production ready

when it is in a *good enough* shape

for the user to serve real-world traffic”

“Your offering is production ready when it slightly exceeds your customer’s expectations in a way that allows for business growth”

-- Carter Morgan, Google (@_askcarter)

It’s all about tradeoffs (!!)

Okay, so what does that mean
in terms of technical work items?

Production-ready cluster?

1. The cluster is *reasonably* secure

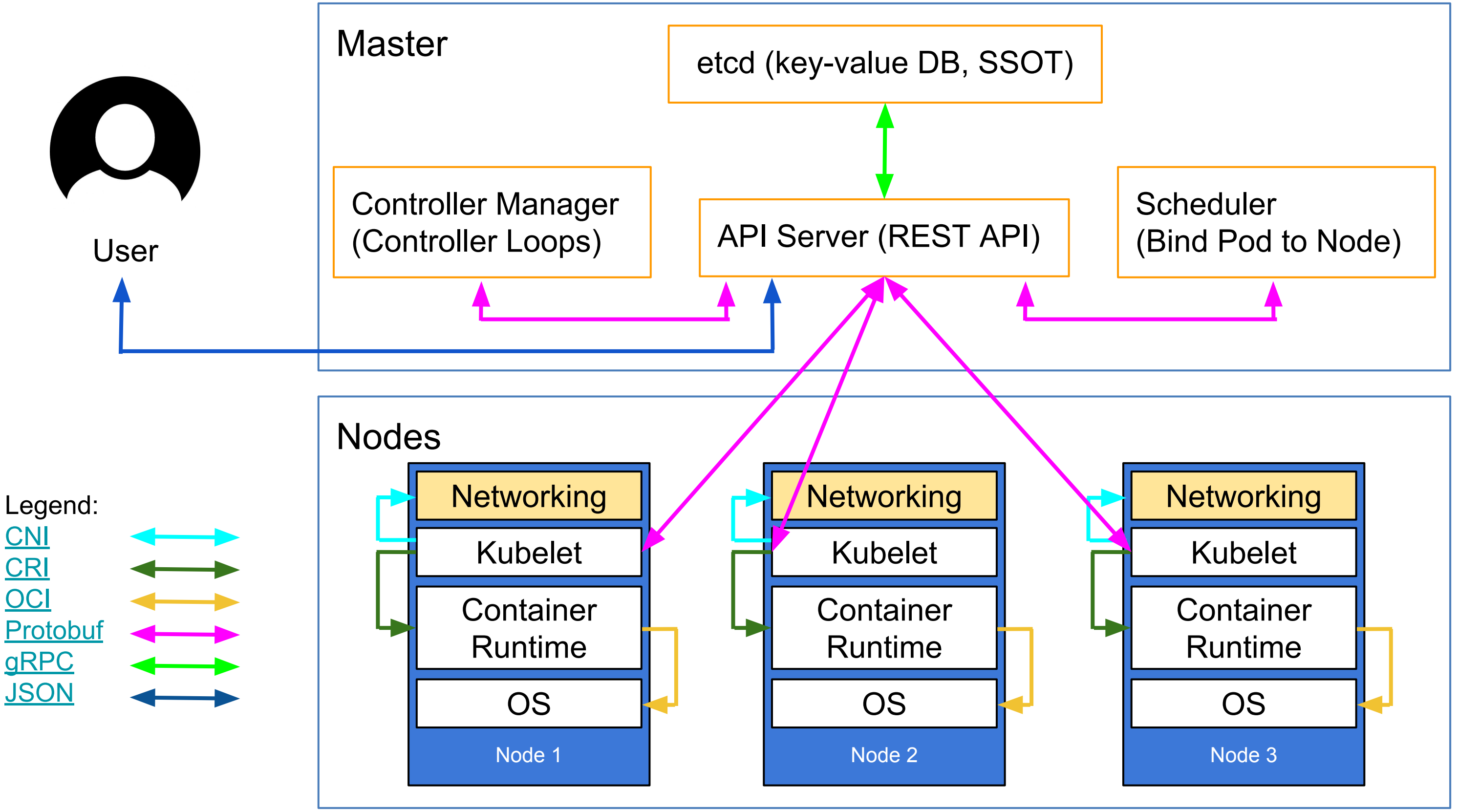
2. The cluster components are *highly available enough* for the user's needs

3. All elements in the cluster are *declaratively controlled*

4. Changes to the cluster state can be *safely applied* (upgrades/rollbacks)

5. The cluster passes as many end-to-end tests as possible

Kubernetes' high-level component architecture



What about “high availability”?

1. ***Instances (>=1) of a component can fail without causing the cluster to fail***
2. ***Machines (>=1) in the cluster can fail without causing the cluster to fail***

More about this in section III.

KubeCon | CloudNativeCon

Europe 2018

II. Securing Kubernetes

Things to keep in mind

1. TLS-secured communication everywhere!

a. Certificates/identities should be ***rotatable***

b. Use a ***separate CA*** for etcd

c. Use the Certificates/CSR API, with an ***external key signer*** if possible

2. API Authentication and Authorization

a. Disable anonymous authentication and localhost:8080
b. Enforce the RBAC and Node authorizers

3. Lock down the kubelets in the cluster

a. Each kubelet should have its *unique identity*

b. Disable the readonly port (10255) & public (!) cAdvisor port (4194)

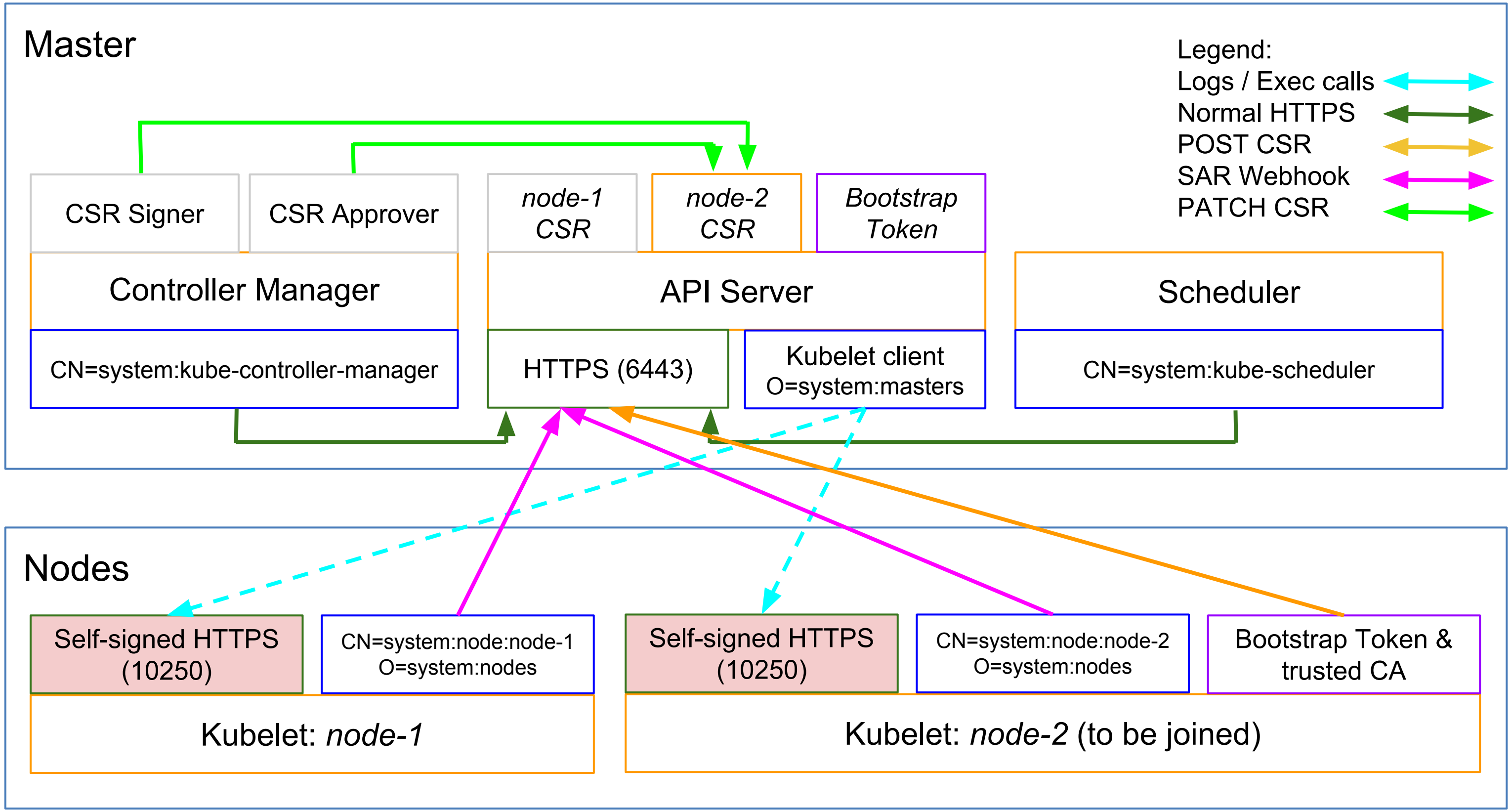
4. Be careful with the Dashboard and Helm

a. ***Don't*** give them *cluster-admin* power, then it's very easy to escalate privileges
b. The security of the dashboard has improved since v1.7.0
    i. The dashboard now has a login screen and delegates privileges
c. Specify the exact operations *tiller* may perform with RBAC
d. Secure the Helm <-> Tiller communication with TLS certificates

5. Deny by default -- best practices security-wise

a. Deny-all with RBAC
b. Deny-all with NetworkPolicy
c. Set up a restrictive PodSecurityPolicy as the default

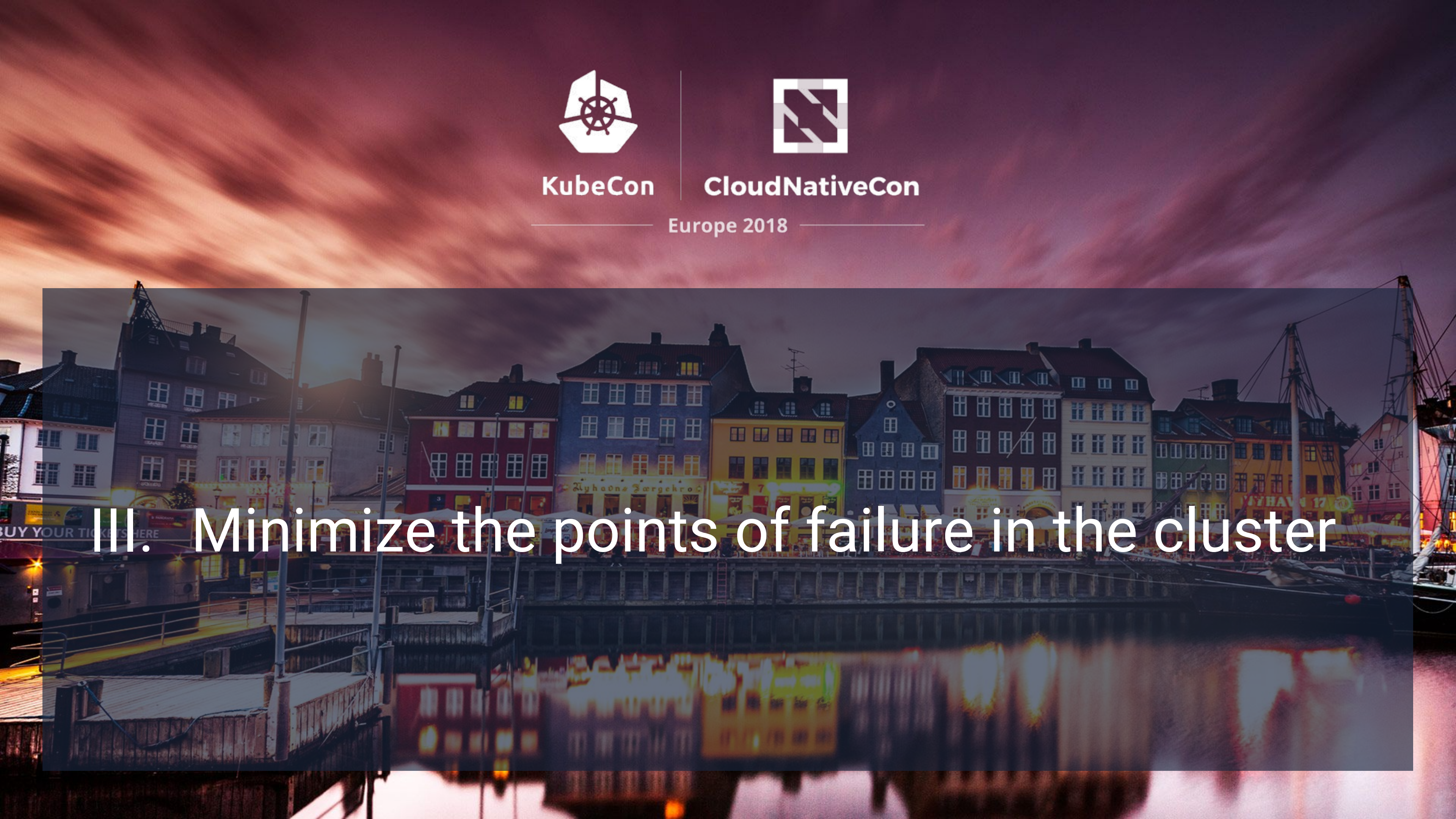
Setting up a dynamic TLS-secured cluster


Master
Legend:
Logs / Exec calls
Normal HTTPS
POST CSR
SAR Webhook
PATCH CSR
CSR Signer
CSR Approver
node-1 CSR
node-2 CSR
Bootstrap Token
Controller Manager
API Server
Scheduler
CN=system:kube-controller-manager
HTTPS (6443)
Kubelet client O=system:masters
CN=system:kube-scheduler
Nodes
Self-signed HTTPS (10250)
CN=system:node:node-1 O=system:nodes
Self-signed HTTPS (10250)
CN=system:node:node-2 O=system:nodes
Bootstrap Token & trusted CA
Kubelet: node-1
Kubelet: node-2 (to be joined)

CSR=Certificate Signing Request, SAR=Subject Access Review

More information about Kubernetes security

1. Use https://github.com/aquasecurity/kube-bench
2. Official docs: Best Practices for Securing a Kubernetes Cluster
3. Hacking and Hardening Kubernetes Clusters by Example [I] - Brad Geesaman



KubeCon CloudNativeCon

Europe 2018

III. Minimize the points of failure in the cluster

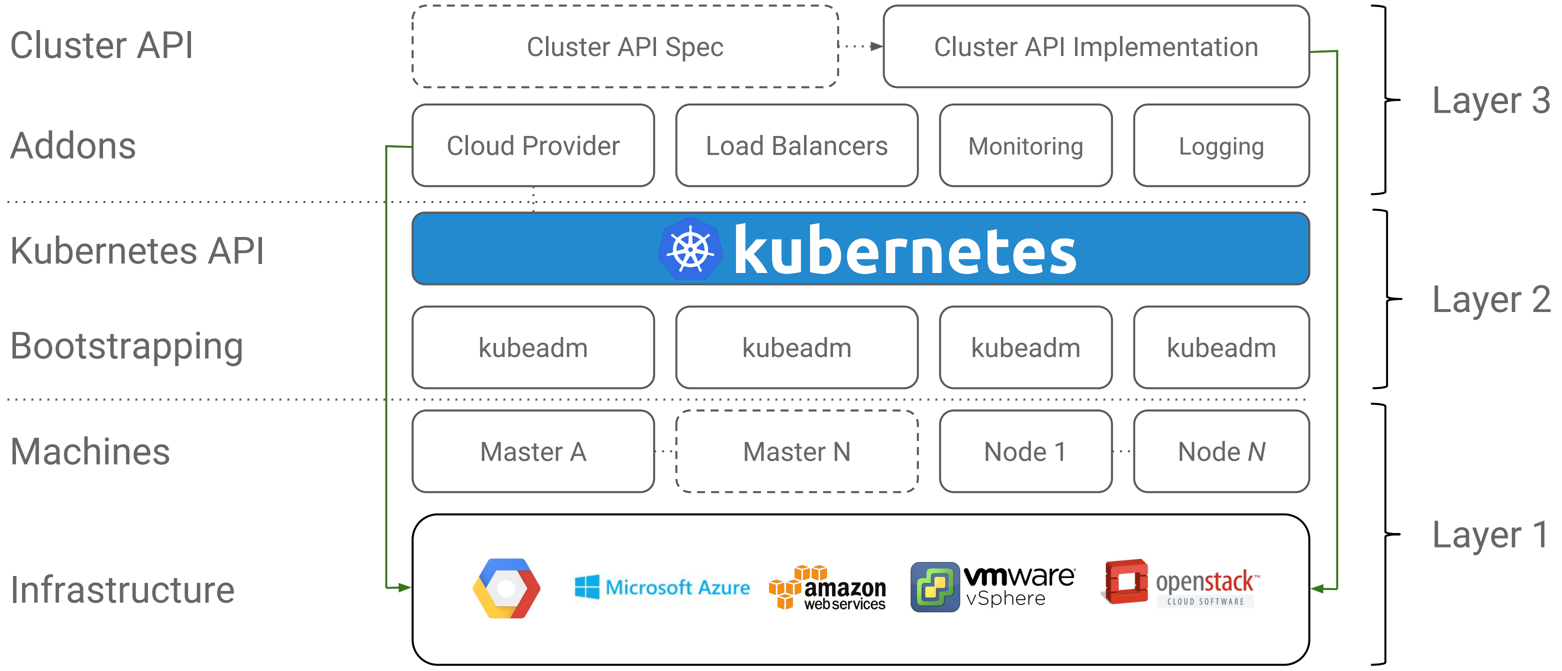
Key design takeaways for kubeadm

- kubeadm’s task is to set up a **best-practice cluster** for each *minor version*
- The user experience should be *simple*, and the cluster reasonably *secure*
- kubeadm’s scope is limited; intended to be a **building block**
    - Only ever deals with the local filesystem and the Kubernetes API
    - Agnostic to ***how exactly*** the kubelet is run
    - Setting up or favoring a specific CNI network is **out of scope**
- Composable architecture with everything divided into **phases**

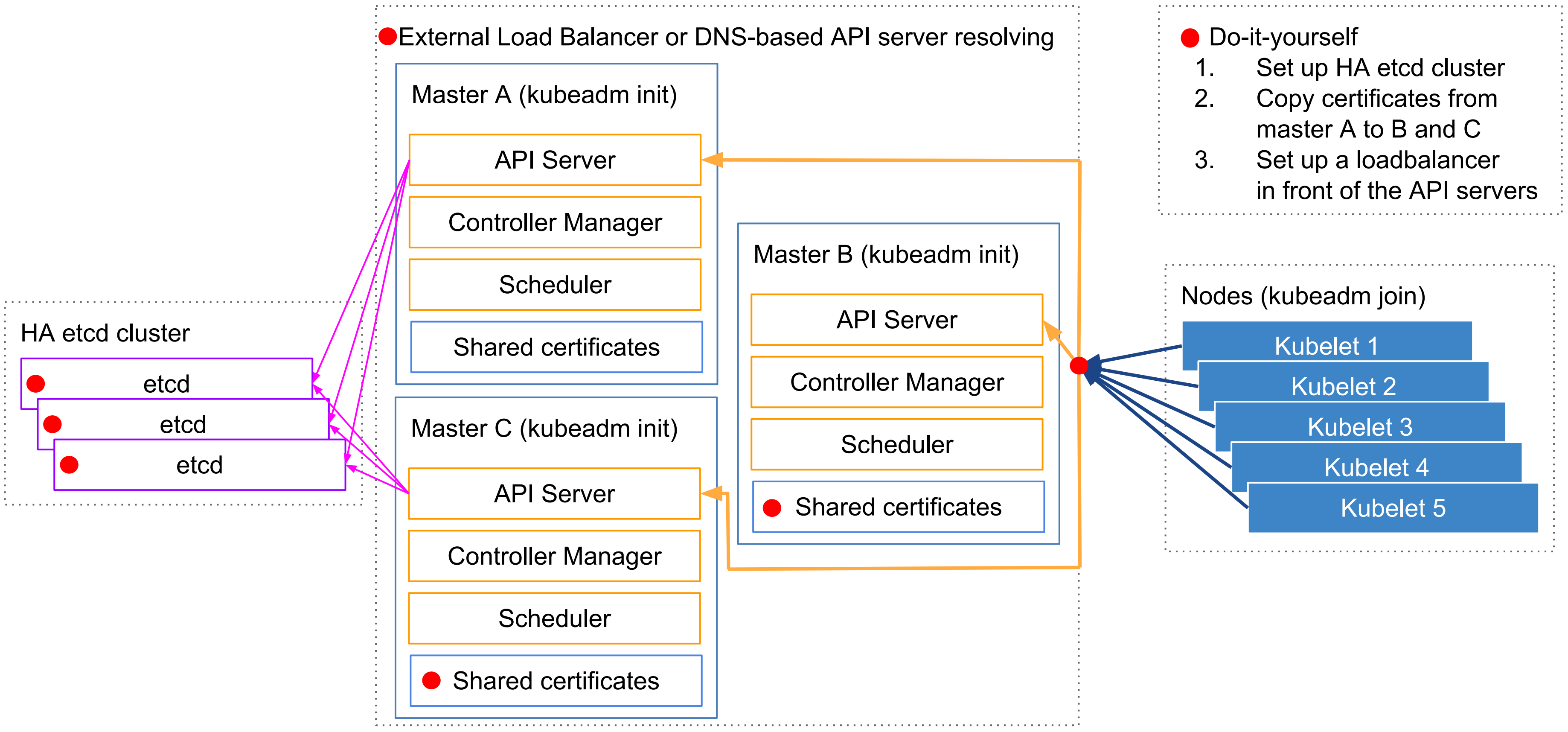
Audience: build-your-first-own-cluster users & higher-level tools like *kops & kubicorn*

What is kubeadm and why should I care?

= A tool that sets up a minimum viable, best-practice Kubernetes cluster



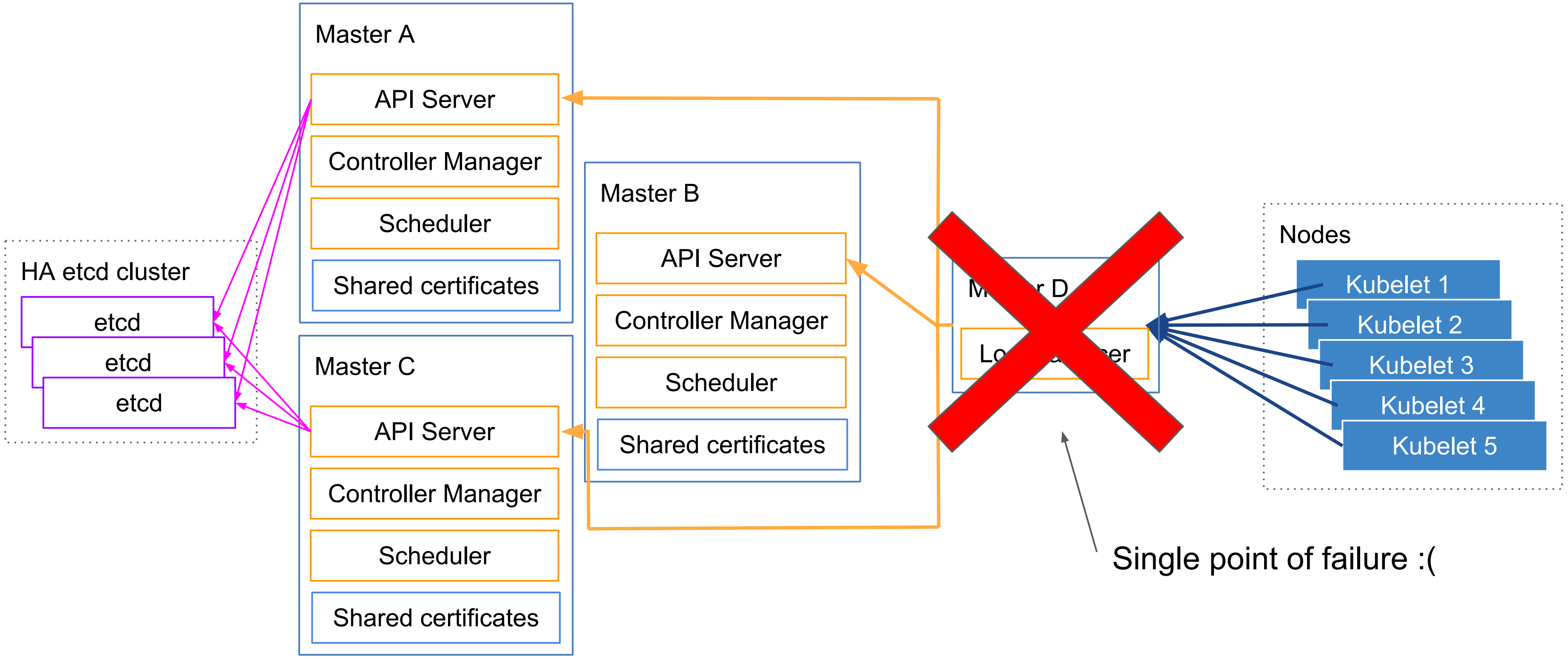
How achieve HA with kubeadm today?



● Do-it-yourself

1. Set up HA etcd cluster
2. Copy certificates from master A to B and C
3. Set up a loadbalancer in front of the API servers

Is this cluster setup highly-available? No



Other things to keep in mind with a HA cluster

1. Remember to keep the kube-dns replicas >= 1, and use Pod anti-affinity
2. Many certificates need to be identical across masters
   a. e.g. the ServiceAccount signing private key for the controller-manager
   b. => Needs to be rotated for all instances at the same time
3. Monitoring the cluster components becomes increasingly more important with a HA cluster that is expected to have a high SLO
   a. You can for example use Prometheus and kube-state-metrics as a starting point

“Monitor it so you know when it fails

before your customers do”

-- Justin Santa Barbara, Google (@justinsb)

KubeCon | CloudNativeCon

Europe 2018

IV. Declarative cluster control with the Cluster API

Manage clusters more like applications

What’s the Cluster API?

- A declarative way to create, configure, and manage a cluster
  - apiVersion: "cluster-api.k8s.io/v1alpha1"
  - kind: Cluster

- Controllers will reconcile desired vs. actual state
  - These could run inside or outside the cluster

- Cloud Providers will implement support for their IaaS
  - GCE, AWS, Azure, Digital Ocean, Terraform and Docker Machine, etc.

- Port existing tools to target Cluster API
  - Cluster upgrades, auto repair, cluster autoscaler

“GitOps” for your cluster with the Cluster API

1. With Kubernetes we manage our applications declaratively
   a. Why don’t we (in some cases) do that for the clusters as well?
2. With the Cluster API, we can declaratively define what the cluster should look like
   a. The installer tools will then consume this “standard” API and act on it
   b. These API types can be stored in a CRD or on disk

```yaml
apiVersion: cluster.k8s.io/v1alpha1
kind: MachineSet
metadata:
  name: my-first-machineset
spec:
  replicas: 3
  selector:
    matchLabels:
      foo: bar
  template:
    metadata:
      labels:
        foo: bar
    spec:
      providerConfig:
        value:
          apiVersion: "gceproviderconfig/v1alpha1"
          kind: "GCEProviderConfig"
          zone: "us-central1-f"
          machineType: "n1-standard-1"
          image: "ubuntu-1604-lts"
      versions:
        kubelet: 1.10.2
        containerRuntime:
          name: docker
          version: 1.12.0
```

Recap

1. Identify the needs of your business
    a. How much money and effort do you want to put into HA & security?

2. High Availability != multiple masters
    a. Multiple masters are a requirement for high availability

3. Pay attention to the certificate identities for your components
    a. And make sure you lock things down well with RBAC, disable unnecessary ports, etc.

4. Declarative control over your cluster is better than imperative
    a. The Cluster API (still alpha) and the GitOps models might be worth checking out

Thank you!

@luxas on Github
@kubernetesonarm on Twitter
lucas@luxaslabs.com

Related resources (in no particular order)

1. https://5pi.de/2017/12/15/production-grade-kubernetes/
2. https://youtu.be/PXJu8ujNEmU
3. https://thenewstack.io/ebooks/kubernetes/state-of-kubernetes-ecosystem/
4. https://kccncna17.sched.com/event/CU5x/101-ways-to-crash-your-cluster-i-marius-grigoriu-emmanuel-gomez-nordstrom
5. https://kccncna17.sched.com/event/CU6H/certifik8s-all-you-need-to-know-about-certificates-in-kubernetes-i-alexander-brand-apprenda
6. https://kccncna17.sched.com/event/CU86/shipping-in-pirate-infested-waters-practical-attack-and-defense-in-kubernetes-a-greg-castle-cj-cullen-google
7. https://kccncna17.sched.com/event/CU6z/hacking-and-hardening-kubernetes-clusters-by-example-i-brad-geesaman-symantec
8. https://kccncna17.sched.com/event/CUFK/keynote-kubernetes-at-github-jesse-newland-principal-site-reliability-engineer-github
9. https://kccncna17.sched.com/event/CU8b/what-happens-when-something-goes-wrong-on-kubernetes-reliability-i-marek-grabowski-tina-zhang-google
10. https://kccncna17.sched.com/event/CU64/automating-and-testing-production-ready-kubernetes-clusters-in-the-public-cloud-ron-lipke-gannetusa-today-network
11. https://stripe.com/blog/operating-kubernetes
12. https://blog.envoyproxy.io/introduction-to-modern-network-load-balancing-and-proxying-a57f6ff80236
13. https://jvns.ca/blog/2017/10/10/operating-a-kubernetes-network/
14. https://acotten.com/post/kube17-security
15. https://applatix.com/making-kubernetes-production-ready/
16. https://www.aquasec.com/wiki/display/containers/Kubernetes+in+Production
17. https://www.weave.works/blog/provisioning-lifecycle-production-ready-kubernetes-cluster/
18. https://www.weave.works/blog/demystifying-production-ready-apps-on-kubernetes-with-carter-morgan
19. https://www.slideshare.net/gn00023040/all-the-troubles-you-get-into-when-setting-up-a-production-ready-kubernetes-cluster
20. https://www.slideshare.net/gn00023040/a-million-ways-of-deploying-a-kubernetes-cluster
21. https://blog.sophaskins.net/blog/misadventures-with-kube-dns/