KubeCon | CloudNativeCon

North America 2018

How we survived our first PCI/HIPPA compliant check with Kubernetes

Nav

Me

Travis Jeppson

Director of Engineering at Nav

Twitter: @stmpy

Keybase: @stmpy

LinkedIn: linkedin.com/in/stmpy

Should I stay and listen?

The Goal: Help other understand the change associated with adopting a Kubernetes workflow, and still abiding to regulations

Which Regulations?

HIPPA

PCI-DSS

*Or a regulation that deals with protecting data

Setting up some common language

Regulations: an authoritative rule dealing with details or procedure

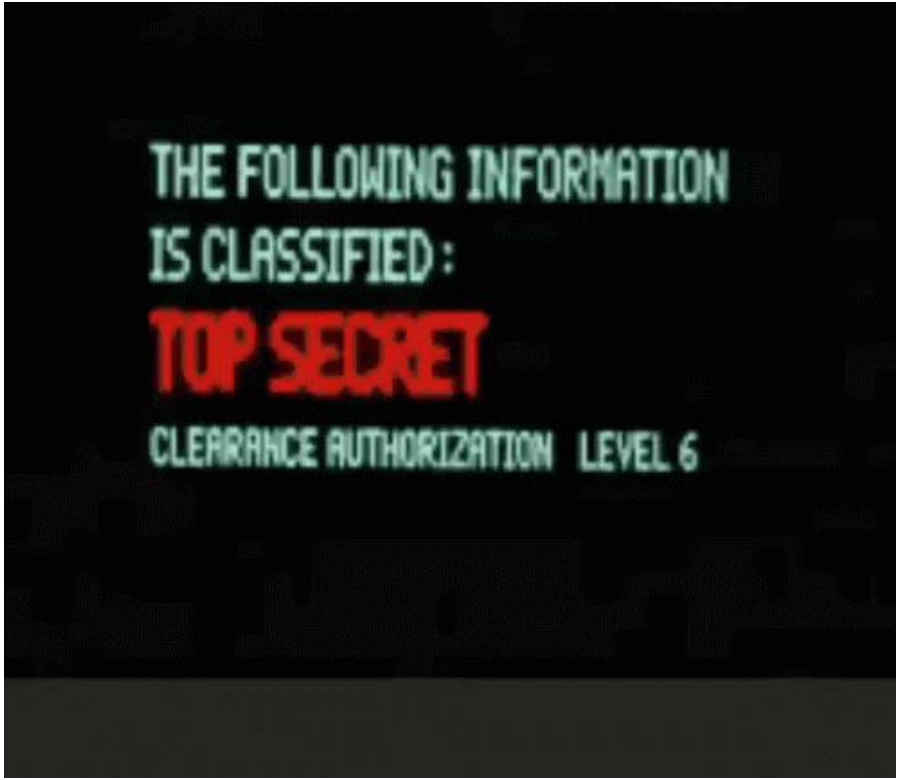
Compliance: the act or process of complying to a desire, demand, proposal, or regimen or to coercion

Standard: something established by authority, custom, or general consent as a model or example

Classification: systematic arrangement in groups or categories according to established criteria

Data Classification

It is important to correctly classify your data



Data Classification (RED)

Social Security Number

Passport Details

Credit Card Information

Drivers License Number

Personal Credit Data

Medical Records

Anything **directly controlled** or that can be used to **individualize** a person

Data Classification (YELLOW)

Birthday

Age

Part of an address

Gender association

Full (Legal) Name

Full Address

The city you were born in

Your favorite school teacher

Where you met your spouse

Your mother's maiden name

The make and model of your first car

Your favorite childhood friend's name

...

Anything that can be used to individualize a person given multiple data points

Data Classification (GREEN)

Unique ID

Encrypted Data

Binary information

Email address

Username

Site interactive data

Basically impossible to individualize a person, or not regulated (public) data

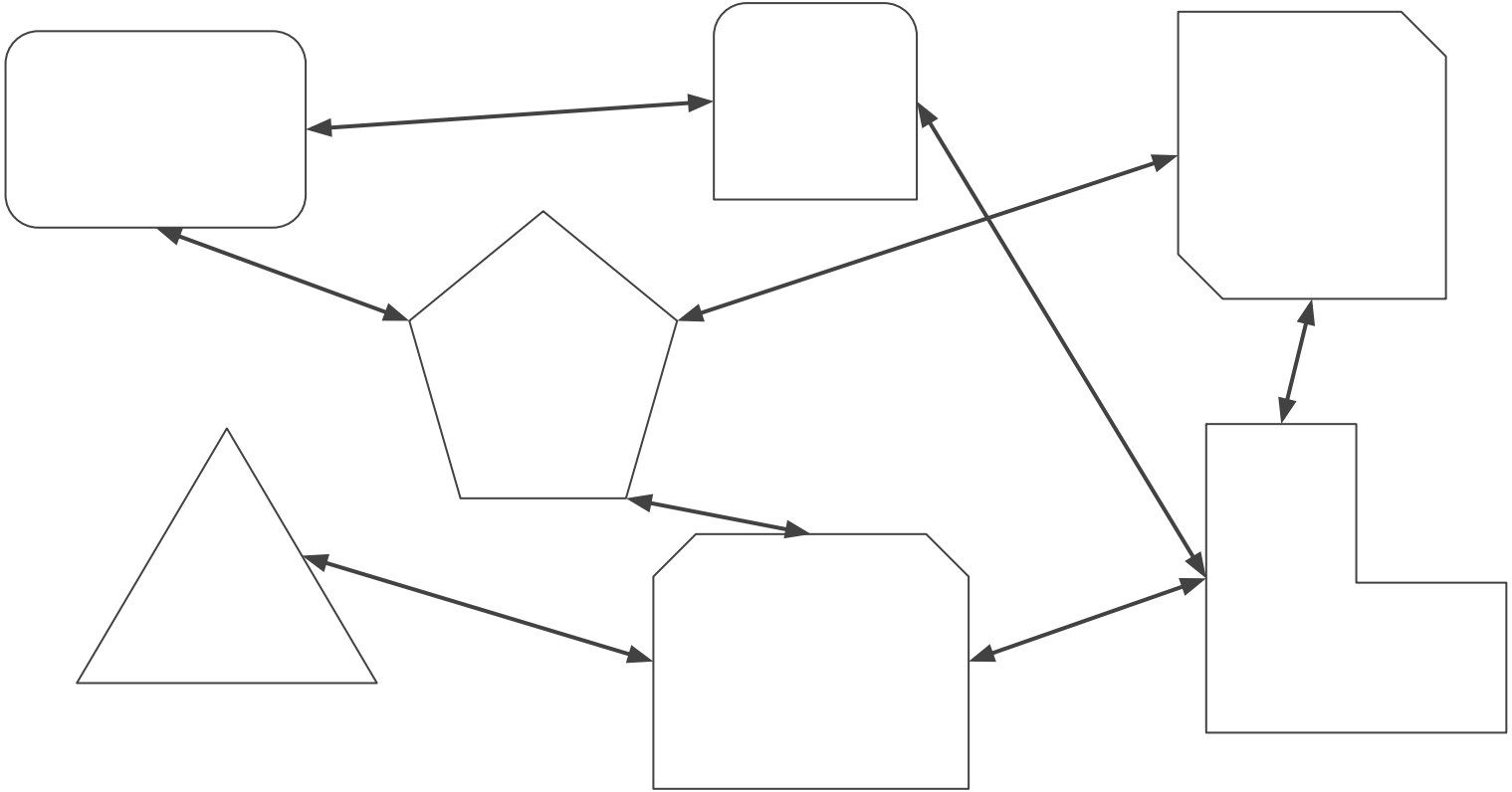
Service Classification follows Data

Service Classification == Data Classification

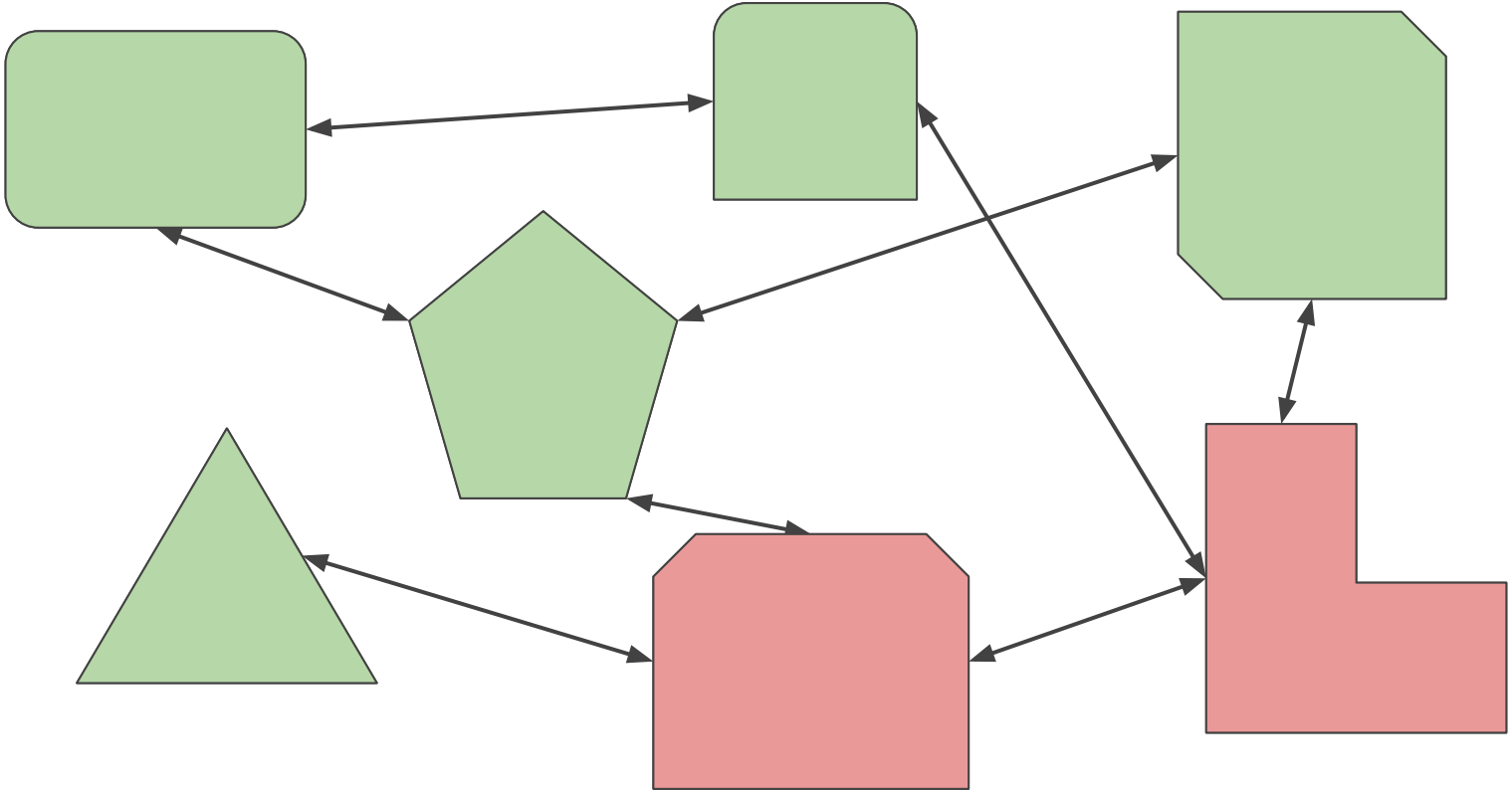
RED == RED

GREEN == GREEN

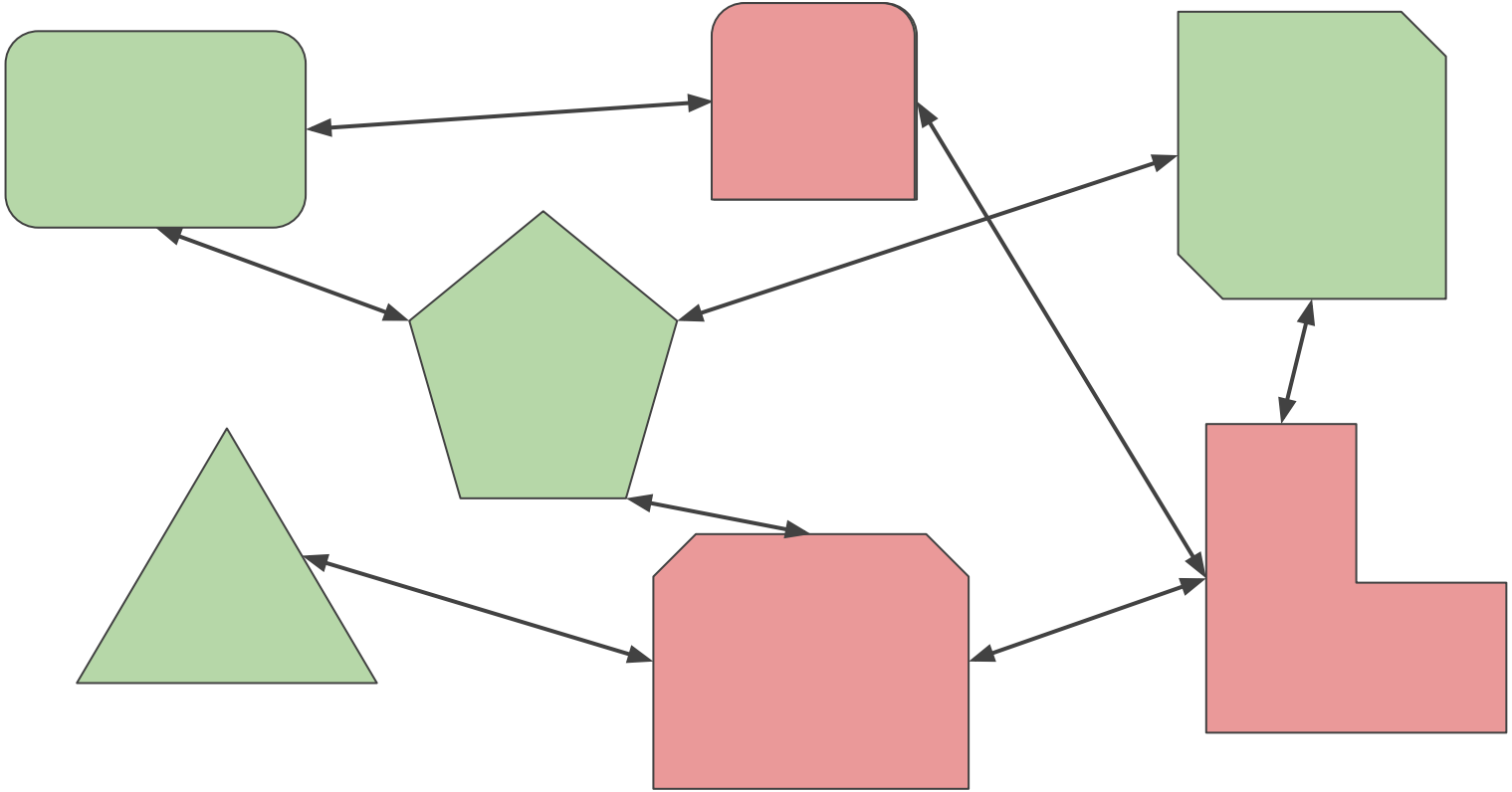
Least Common Denominator



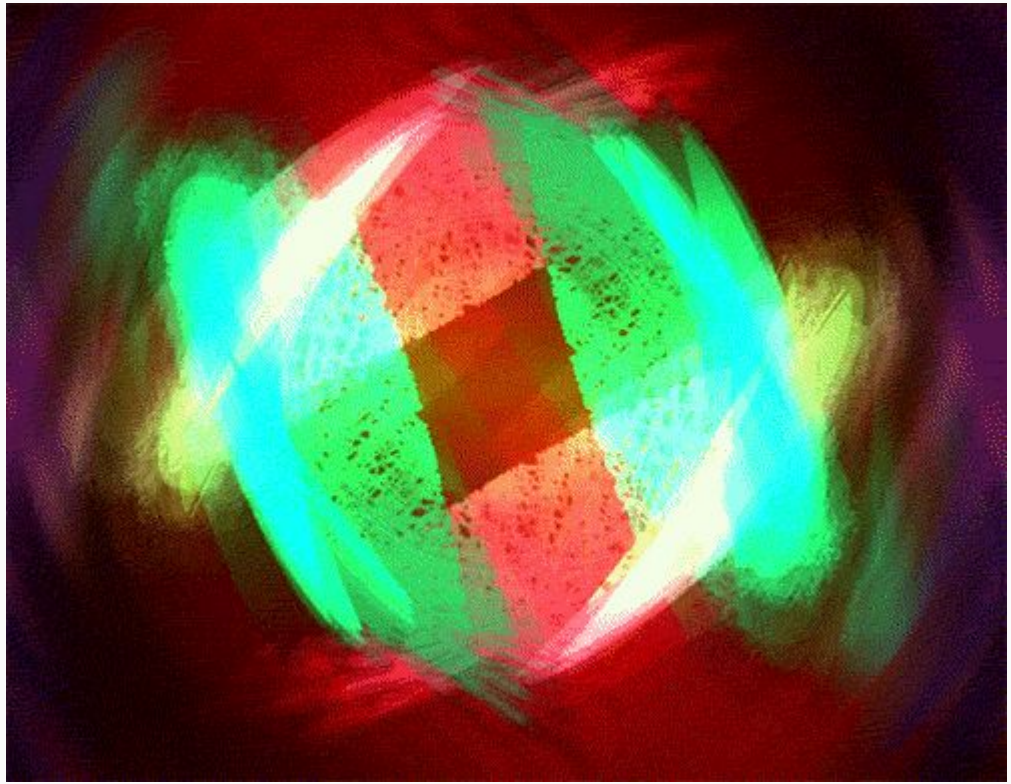
Least Common Denominator



Least Common Denominator



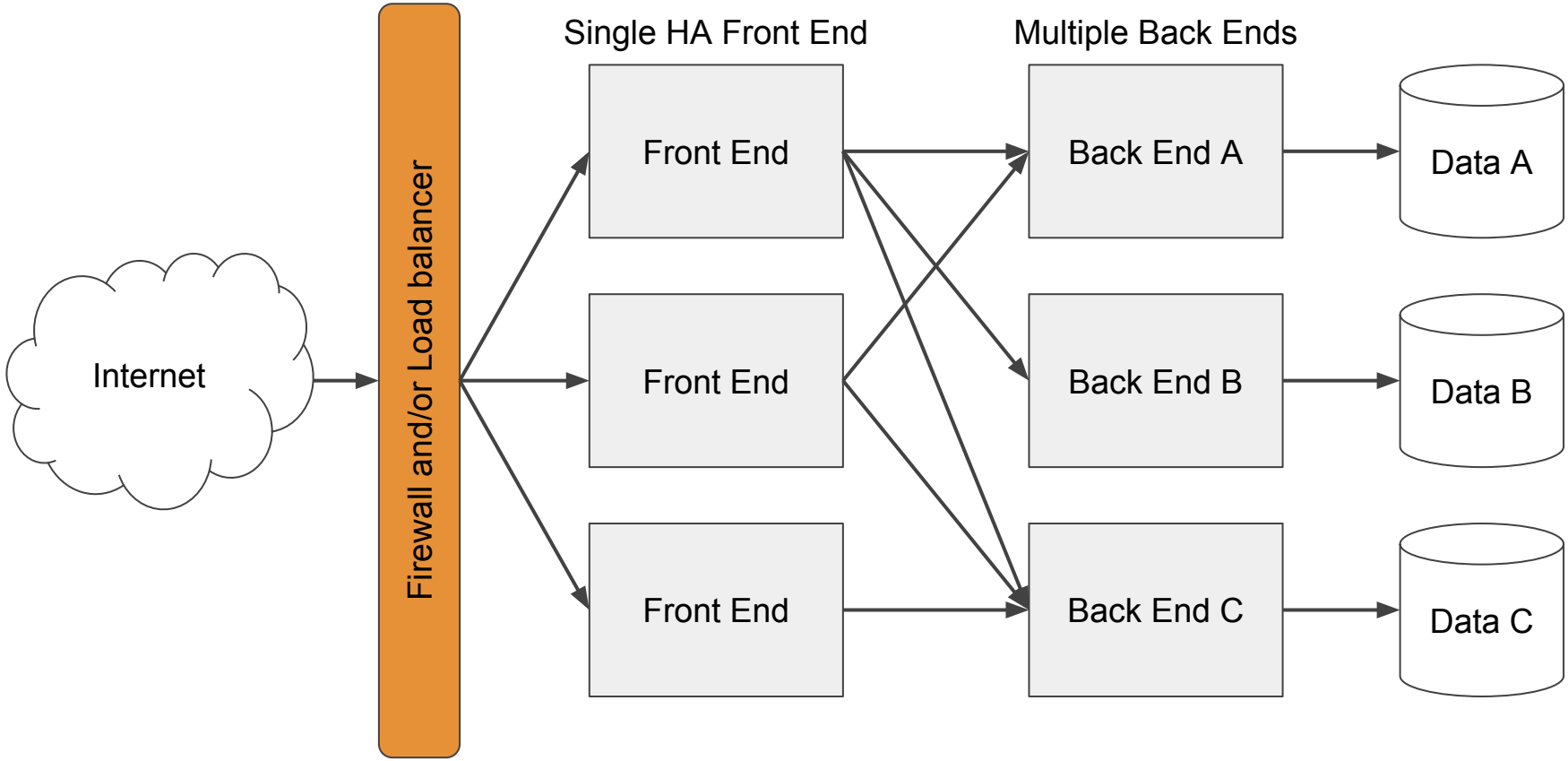
Mixing Green and Red Data



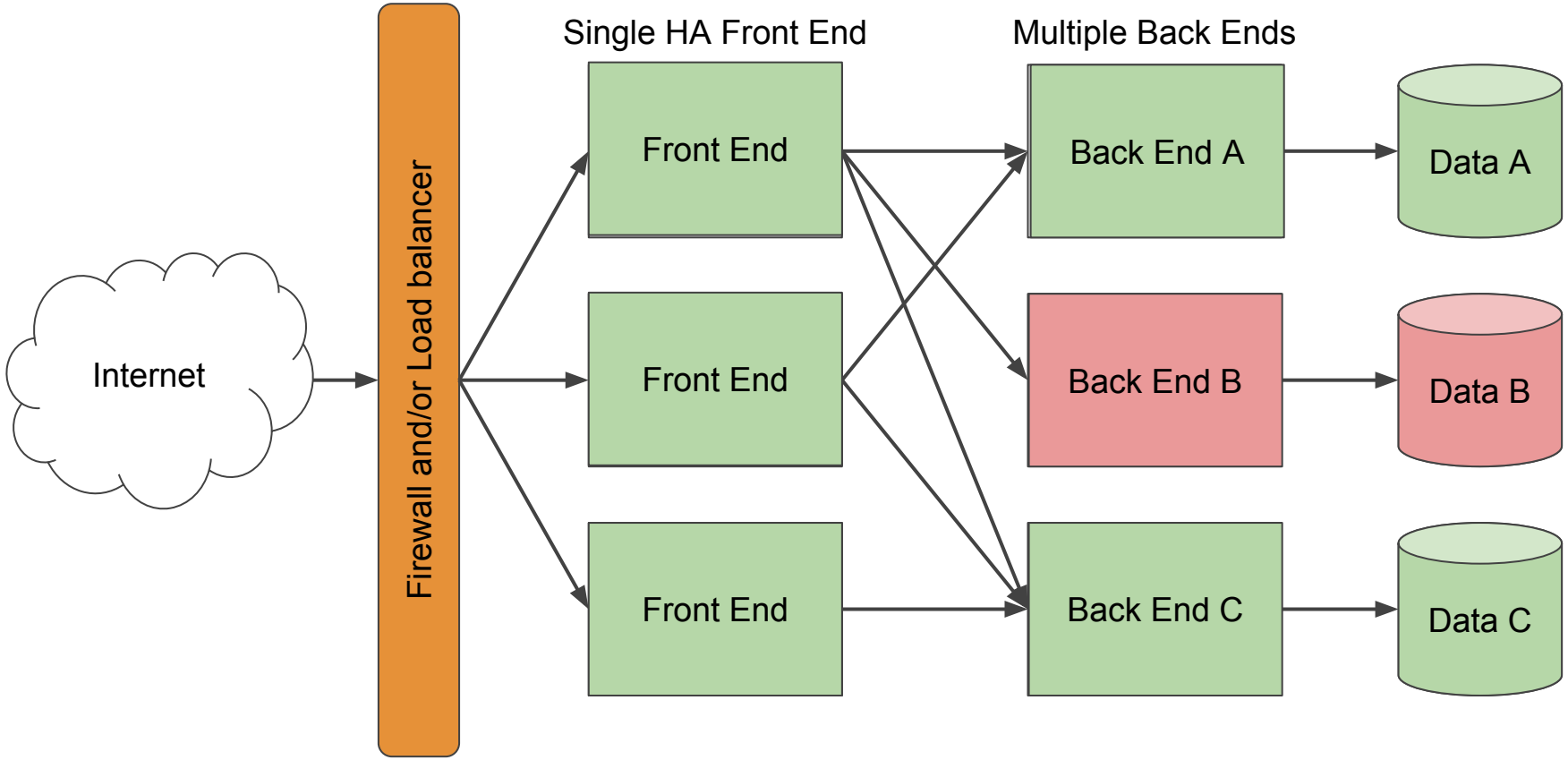
Environment Comparison

How do regulations change for Kubernetes?

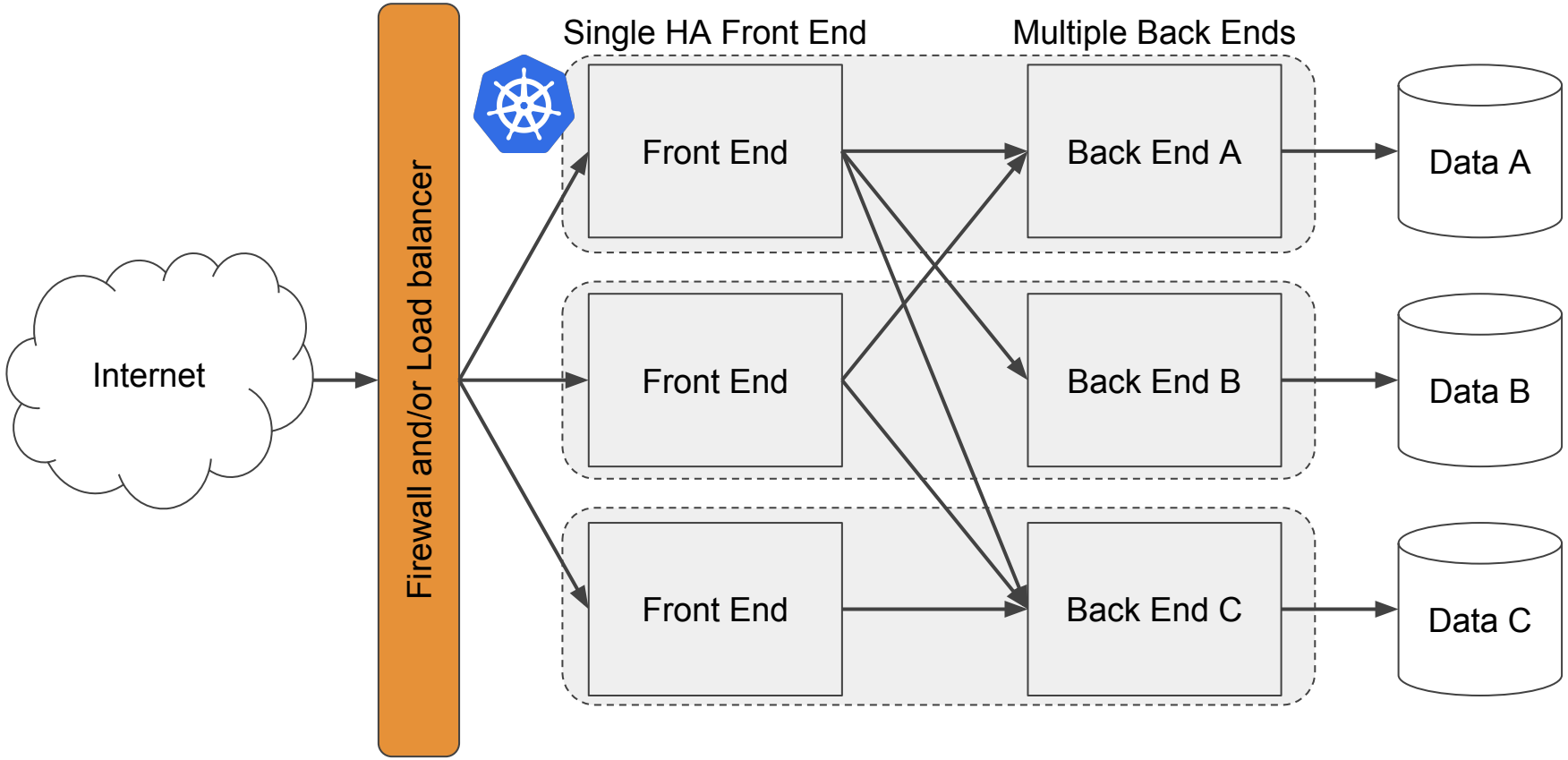
"Traditional" 1:1 - service per machine (vm)



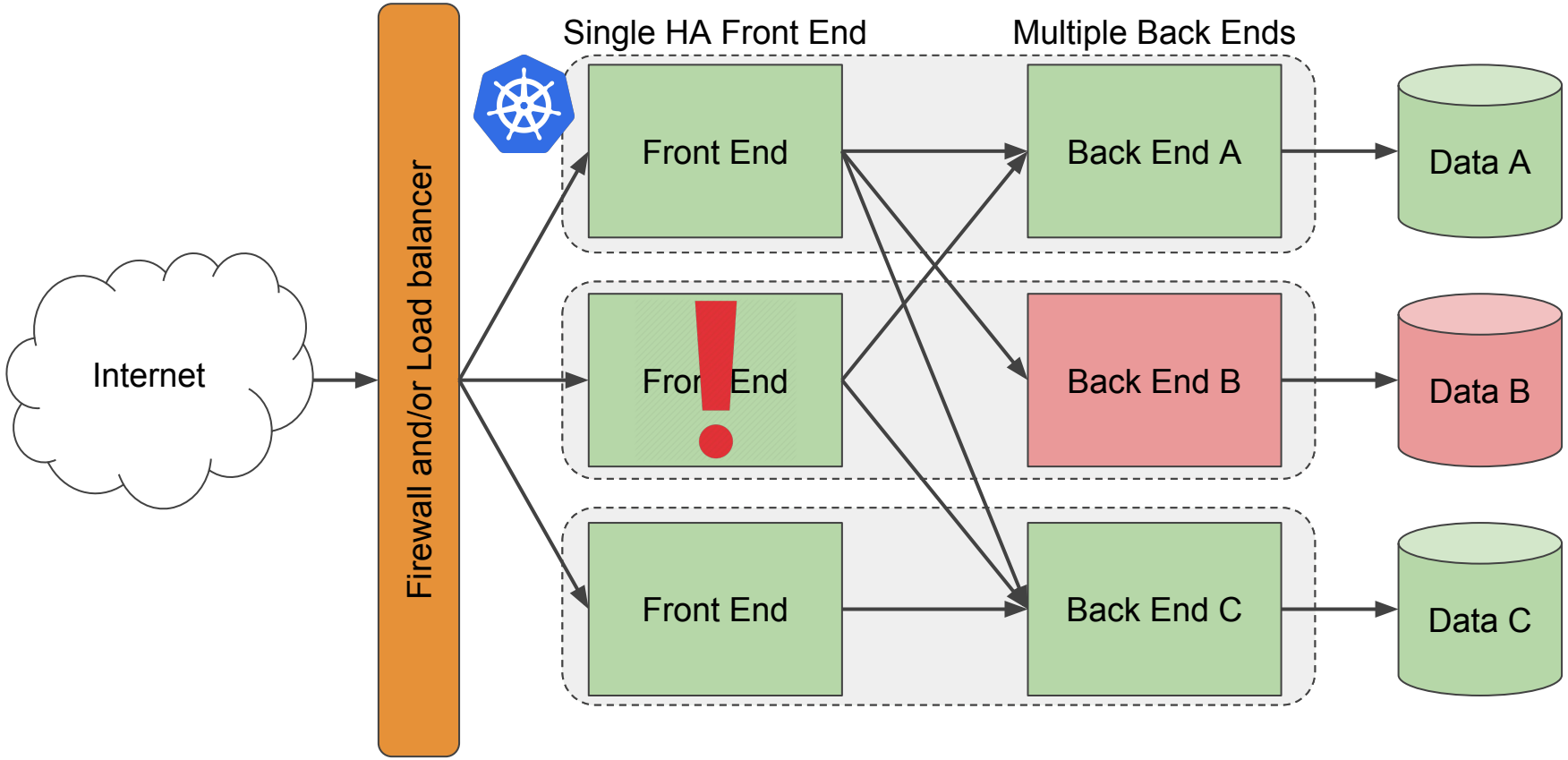
"Traditional" 1:1 - Data Classification



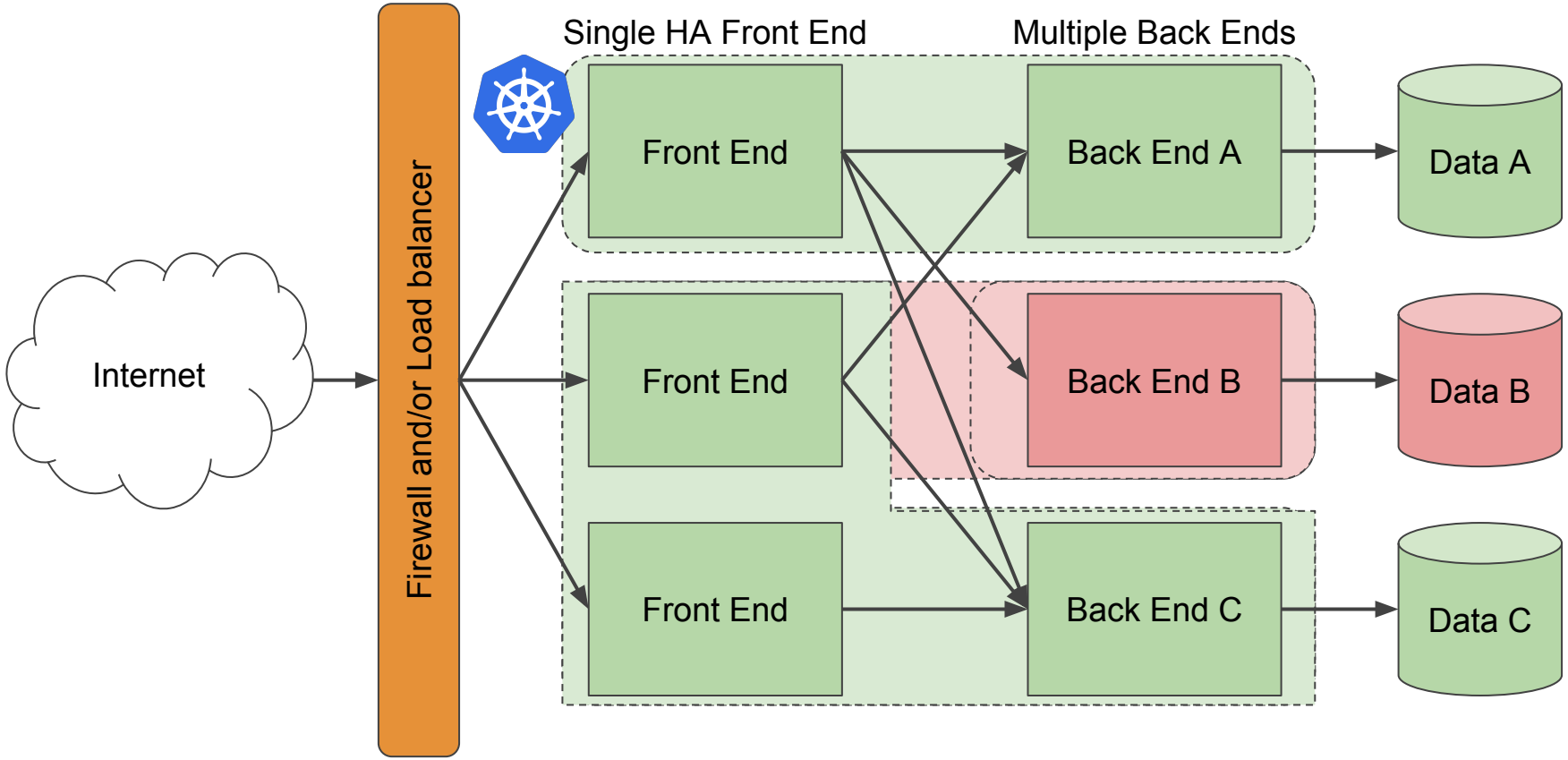
"Distributed" - multiple services per machine



"Distributed" - multiple services per machine



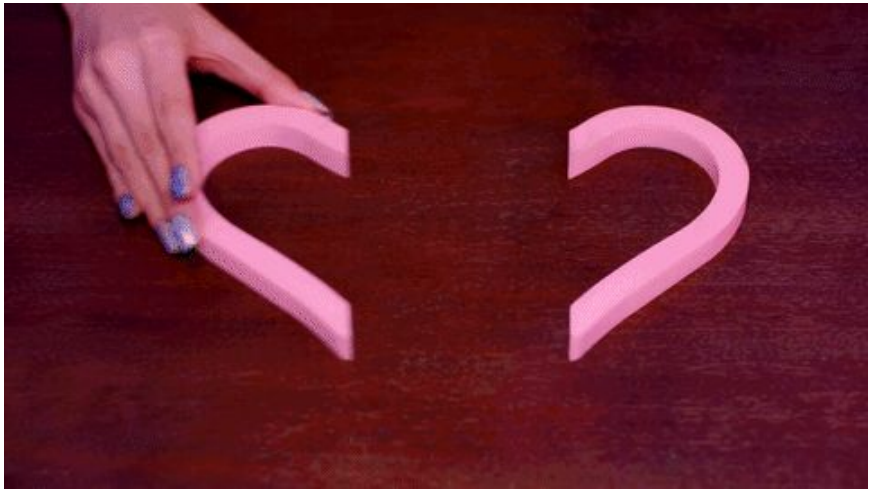
"Classified Distributed" - multiple services per machine



Logical Classification

Taints and Tolerances

https://kubernetes.io/docs/concepts/configuration/taint-and-toleration/



Taints

Applies to a whole node.

```
kubectl taint nodes node1 key=value:NoSchedule
```

places a taint on node `node1`. The taint has key `key`, value `value`, and taint effect `NoSchedule`. This means that no pod will be able to schedule onto `node1` unless it has a matching toleration.

Tolerance

You specify a toleration for a pod in the PodSpec. Both of the following tolerations “match” the taint created by the `kubectl taint`, and thus a pod with either toleration would be able to schedule onto `node1`:

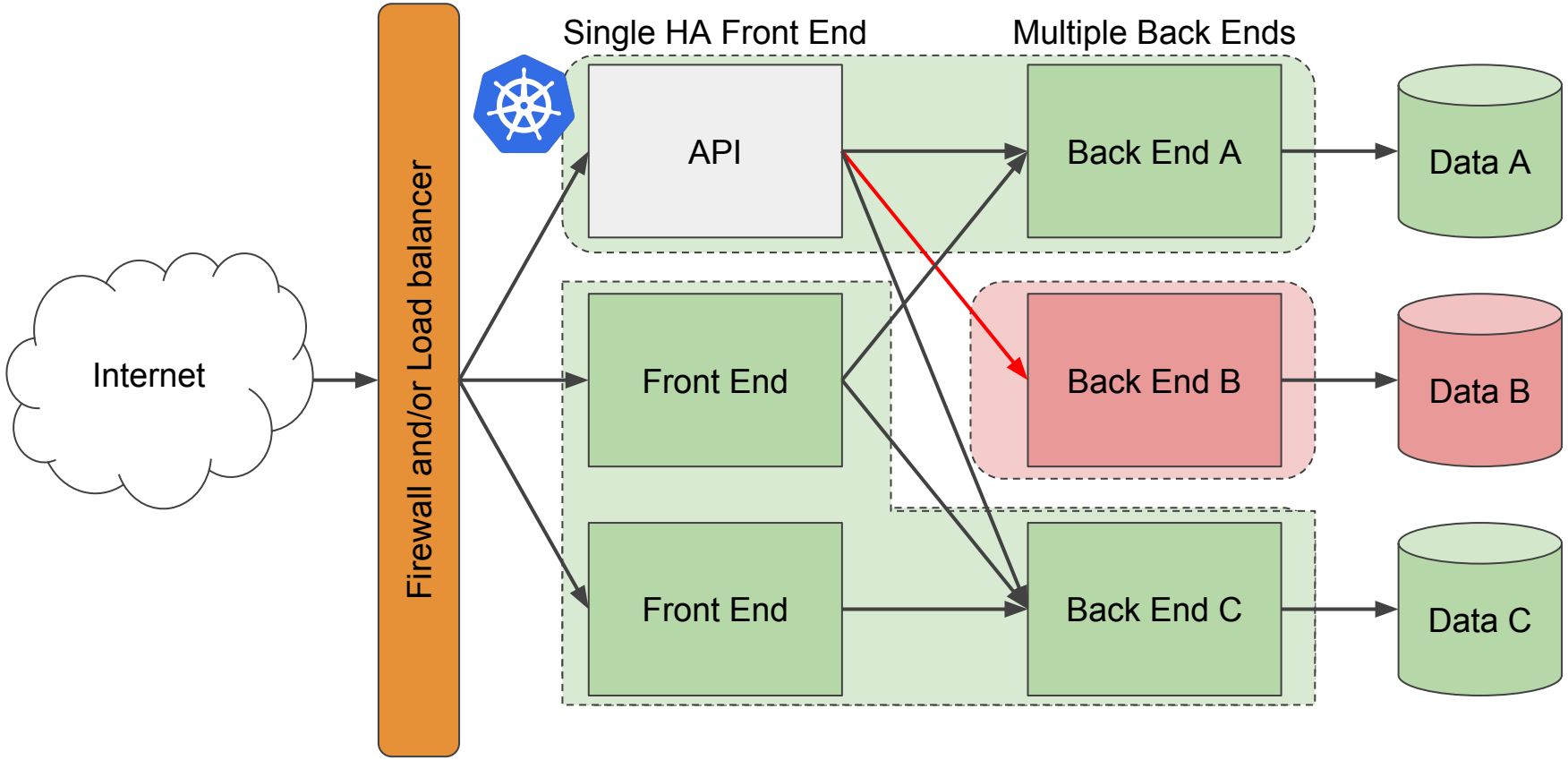
```
tolerations:
- key: "key"
  operator: "Equal"
  value: "value"
  effect: "NoSchedule"

tolerations:
- key: "key"
  operator: "Exists"
  effect: "NoSchedule"
```

Right Here

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', 'echo Hello
Kubernetes! && sleep 3600']
```

Pod-Pod Network Communication



Pod-Pod Network Restrictions

There are multiple ways to handle this, but none of which are natively built into Kubernetes.

NetworkPolicy: with a supporting CNI layer

Cilium - https://cilium.io/ (also includes layer 7 security controls)

Calico - https://www.projectcalico.org/

Weave.net - https://www.weave.works/oss/net/

CNI Networking layers are difficult to replace, you will probably want to work with the vendor to make sure you don't run into any issues

Pod-Pod Network Restrictions

Service Mesh:

Linkerd - https://linkerd.io/

Istio - https://istio.io/

Service Meshes aren't always a "drop in" solution

They do come with more than just network regulations:

- TLS everywhere
- Pod failover
- Lots of metrics

Pod-Pod Network Restrictions

Cloud Native Firewall:

TwistLock - https://www.twistlock.com/

Deployable with containers and `DaemonSets`

(This is the option we ended up using - more on this later)

Virus Protection

Virus Protection in Containers

static (during build) and dynamic (during runtime) scanning are a must!!!!

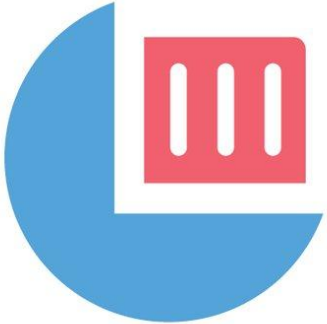
There are a lot of options here: http://lmgtfy.com/?q=container+scanning

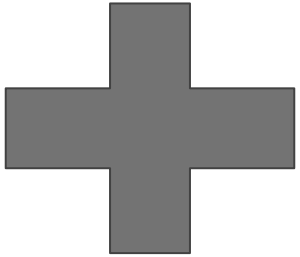
- CoreOS Claire: https://github.com/coreos/clair
- Docker Bench Security: https://github.com/docker/docker-bench-security
- Qualys: https://www.qualys.com/solutions/devsecops/
- Anchore: https://anchore.com/
- Twistlock: https://twistlock.com/
- Sysdig: https://sysdig.com/products/secure/

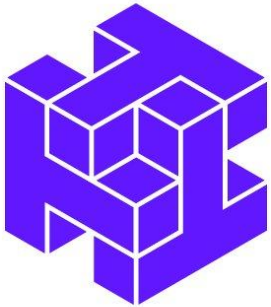
Most of these are platforms that help protect from multiple angles: runtime, build as well as on the node itself.

Virus Protection on Node

This was pretty easy to achieve with Container Linux + container security platform (from previous slide), for us it was Twistlock







Pipeline Practices

Things that have helped us stay secure

"Gold master" Base Image

```
FROM alpine:3.8

RUN apk --no-cache upgrade

RUN apk --no-cache add \
      curl \
      ca-certificates \
      bash \
      shadow \
      jq

COPY *.crt /usr/local/share/ca-certificates/

RUN update-ca-certificates

COPY entrypoint.sh /entrypoint.sh

COPY alpine.gitlog /alpine.gitlog

ENTRYPOINT ["/entrypoint.sh"]
```

Base all other projects off of this image (as much as possible anyway).

This helps immensely when trying to push updates and vulnerability fixes out.

Static Container Image Scanning

```
No CA cert was specified, using insecure connection
Vulnerabilities
---------------
Image    ID    CVE    Package    Version    Severity    Status
-----    --    ---    -------    -------    --------    ------
Vulnerability threshold check results: PASS

Compliance
----------
Image                                                ID                  Severity    Description
-----                                                --                  --------    -----------
registry.nav.engineering/goldmaster/alpine:latest    921e67f2a023fca3    high        Image should be created with a user
```

Set a threshold on the vulnerability/compliance scanning to fail builds if surpassed.

Scan anything that builds and pushes into your container registry.

SAST - Static Application Security Testing

Request to merge `awesome-feature` **into** `master`

Check out branch

Pipeline #18777035 passed for 8805f6cd.

SAST improved on 1 security vulnerability and degraded on 4 security vulnerabilities

Collapse

- ✗ Medium: Cipher with no integrity in src/main/java/com/gitlab/security_products/tests/App.java:29
- ✗ Medium: ECB mode is insecure in src/main/java/com/gitlab/security_products/tests/App.java:29
- ✗ Medium: Predictable pseudorandom number generator in src/main/java/com/gitlab/security_products/tests/App.java:41
- ✓ Medium: Predictable pseudorandom number generator in src/main/java/com/gitlab/security_products/tests/App.java:47

Show complete code vulnerabilities report

Merge ☑ Remove source branch ☑ Squash commits ? Modify commit message

DAST - Dynamic Application Security Testing

Request to merge `add-dast` **into** `master`

Check out branch

Pipeline #58 passed for 7a941f11.

7 DAST alerts detected by analyzing the review app Collapse

- Low (Medium): Absence of Anti-CSRF Tokens
- Low (Medium): X-Content-Type-Options Header Missing
- Low (Medium): Password Autocomplete in Browser
- Low (Medium): Private IP Disclosure
- Informational (Medium): Information Disclosure - Suspicious Comments
- Medium (Medium): Application Error Disclosure
- Medium (Low): HTTP Parameter Override

Merge ☐ Remove source branch ☐ Squash commits Modify commit message

You can merge this merge request manually using the command line

Questions?

Nav