& C

D V P

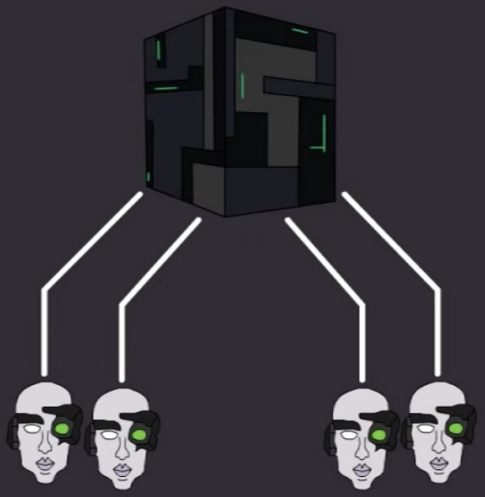
Engineering

Kubernetes securities from built-in features to open source tools

By a,b,c

Contents

About k8s
k8s security
Demos

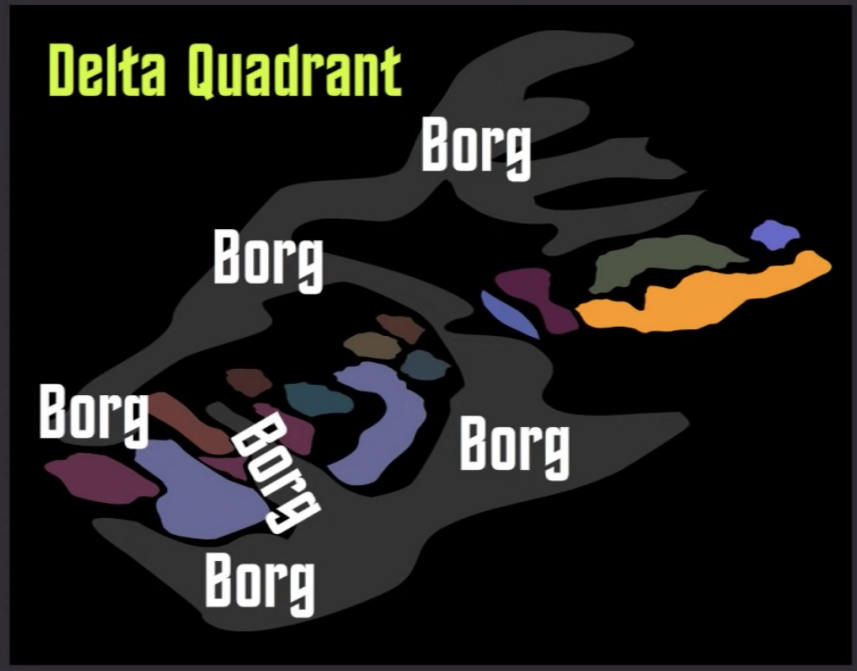


Collective Consciousness

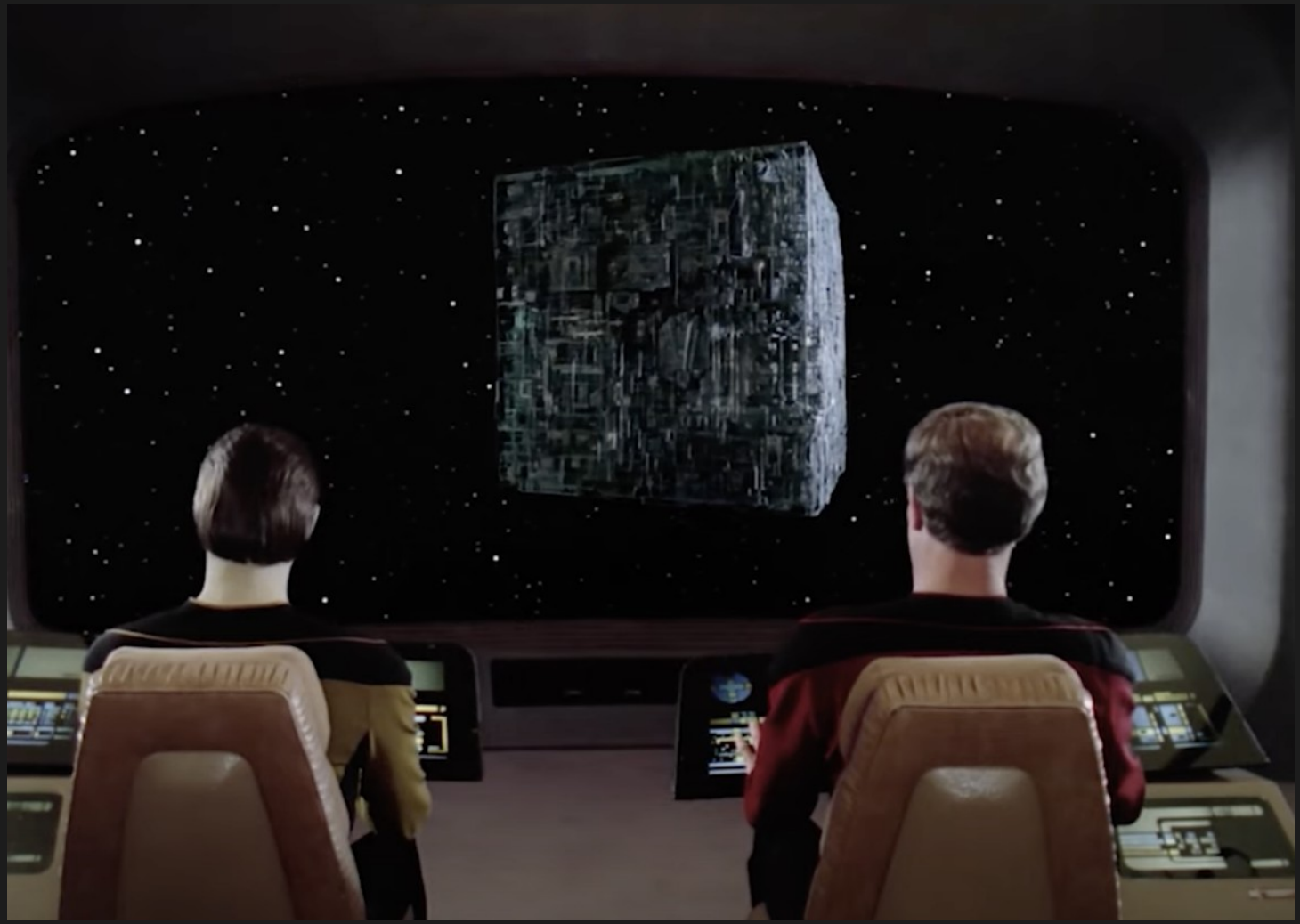
All Borg Minds Linked Together

Shared Thoughts and Memories

Acted as Unified Entity



borg



borg

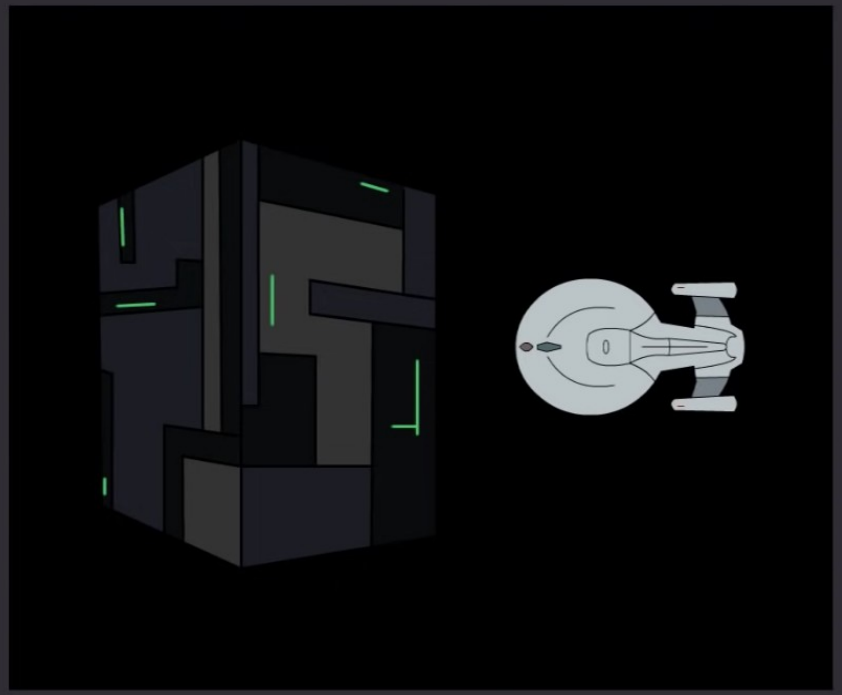
Borg-Voyager Alliance

Voyager's Doctor Created Modified Nanoprobes to Fight Species 8472
Janeway Agreed to Work with the Borg in Exchange for Passage through their Space
Seven of Nine Represented the Borg
Borg Won the War
Voyager Escaped Borg's Betrayal

Seven of Nine
Tertiary Adjunct of Unimatrix Zero One

Borg-Species 8472 War

2373-2374



/ kubernetes-first-commit Public

Notifications Fork 0 Star 1

Code Pull requests Actions Security Insights

first-commit 1 branch 0 tags Go to file Code

jbeda First commit 2c4b3a5 on Jun 7, 2014 1 commit

api	First commit	8 years ago
cmd	First commit	8 years ago
examples/guestbook	First commit	8 years ago
hooks	First commit	8 years ago
pkg	First commit	8 years ago
src	First commit	8 years ago
third_party	First commit	8 years ago
.gitignore	First commit	8 years ago
LICENSE	First commit	8 years ago
README.md	First commit	8 years ago

README.md

Kubernetes

Kubernetes is an open source reference implementation of container cluster management.

Getting started on Google Compute Engine

Prerequisites

1. You need a Google Cloud Platform account with billing enabled. Visit http://cloud.google.com/console for more details
2. You must have Go installed: www.golang.org

About

The first commit of kubernetes/kubernetes. Borrowed from github.com/kubernetes/kubernetes.

Readme
Apache-2.0 license
1 star
1 watching
0 forks

Releases

No releases published

Packages

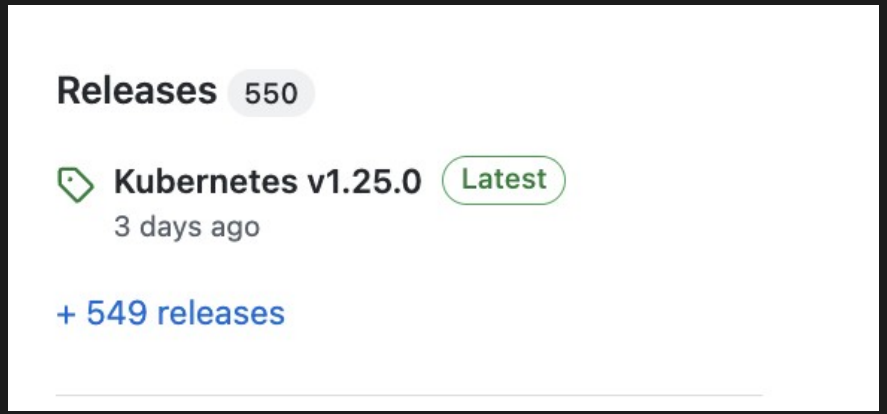
No packages published

Languages

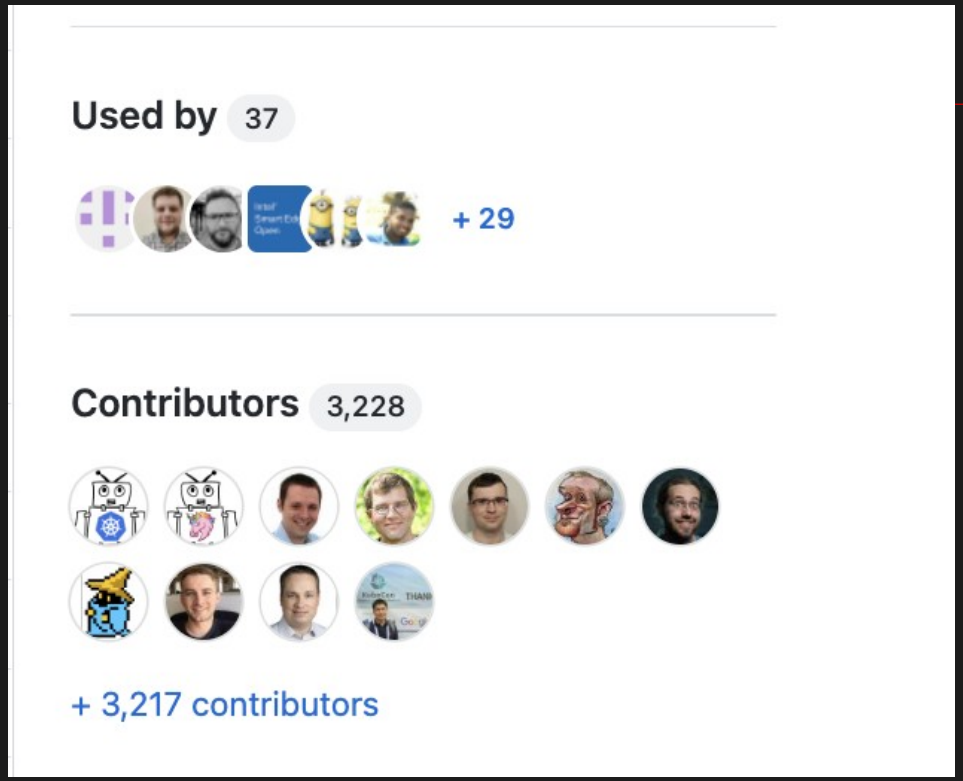
Go 57.5% HTML 25.0%
Shell 12.0% SaltStack 2.6%
RAML 1.6% Python 1.3%

6.7 millions loc

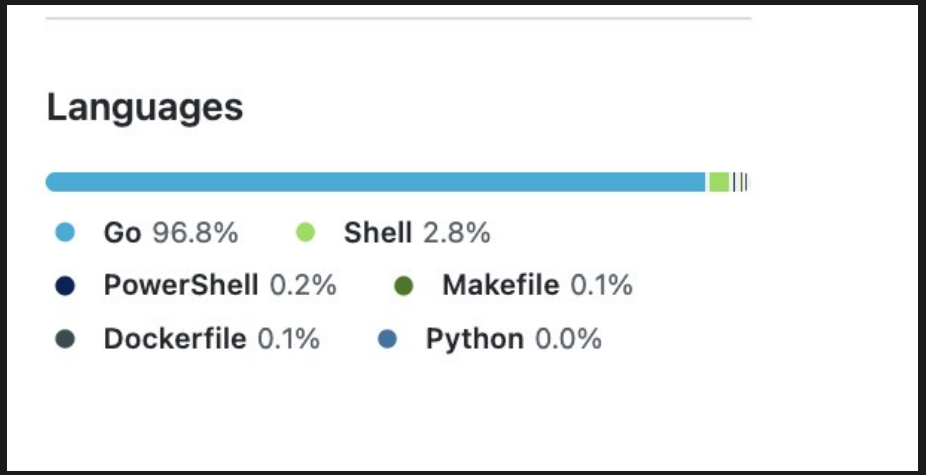
```
bash-5.1$ git grep ^ | wc -l
 6717725
bash-5.1$ pwd;
/Users/user/Documents/k8s/kubernetes
bash-5.1$
```

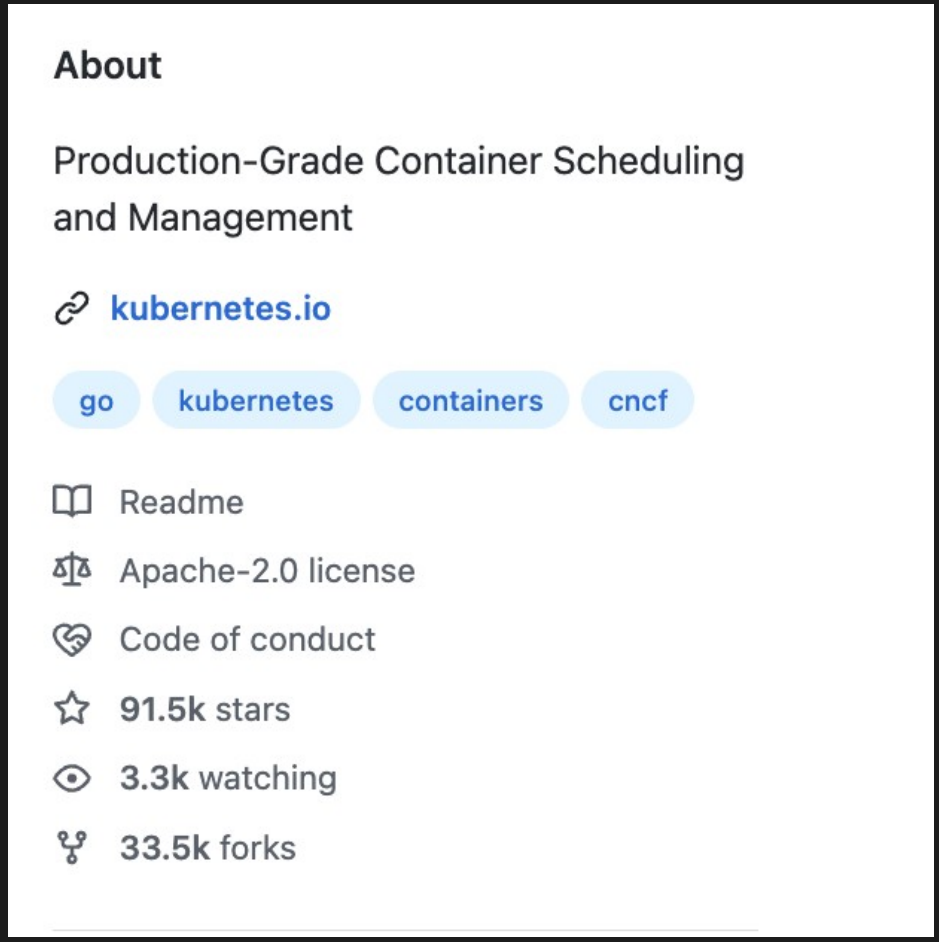


k8s first commit on June 7 2014



k8s first commit on June 7 2014





10 years borg

Large-scale cluster management at Google with Borg

Abhishek Verma† Luis Pedrosa‡ Madhukar Korupolu
David Oppenheimer Eric Tune John Wilkes

Google Inc.

Abstract

Google's Borg system is a cluster manager that runs hundreds of thousands of jobs, from many thousands of different applications, across a number of clusters each with up to tens of thousands of machines.

It achieves high utilization by combining admission control, efficient task-packing, over-commitment, and machine sharing with process-level performance isolation. It supports high-availability applications with runtime features that minimize fault-recovery time, and scheduling policies that reduce the probability of correlated failures. Borg simplifies life for its users by offering a declarative job specification language, name service integration, real-time job monitoring, and tools to analyze and simulate system behavior.

We present a summary of the Borg system architecture and features, important design decisions, a quantitative analysis of some of its policy decisions, and a qualitative examination of lessons learned from a decade of operational experience with it.

1. Introduction

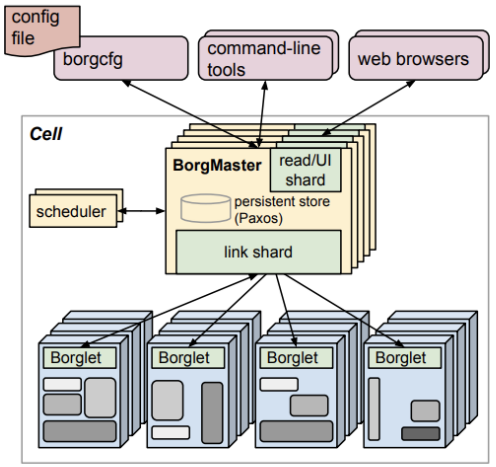


Figure 1: The high-level architecture of Borg. *Only a tiny fraction of the thousands of worker nodes are shown.*

cluding with a set of qualitative observations we have made from operating Borg in production for more than a decade.

10 ye

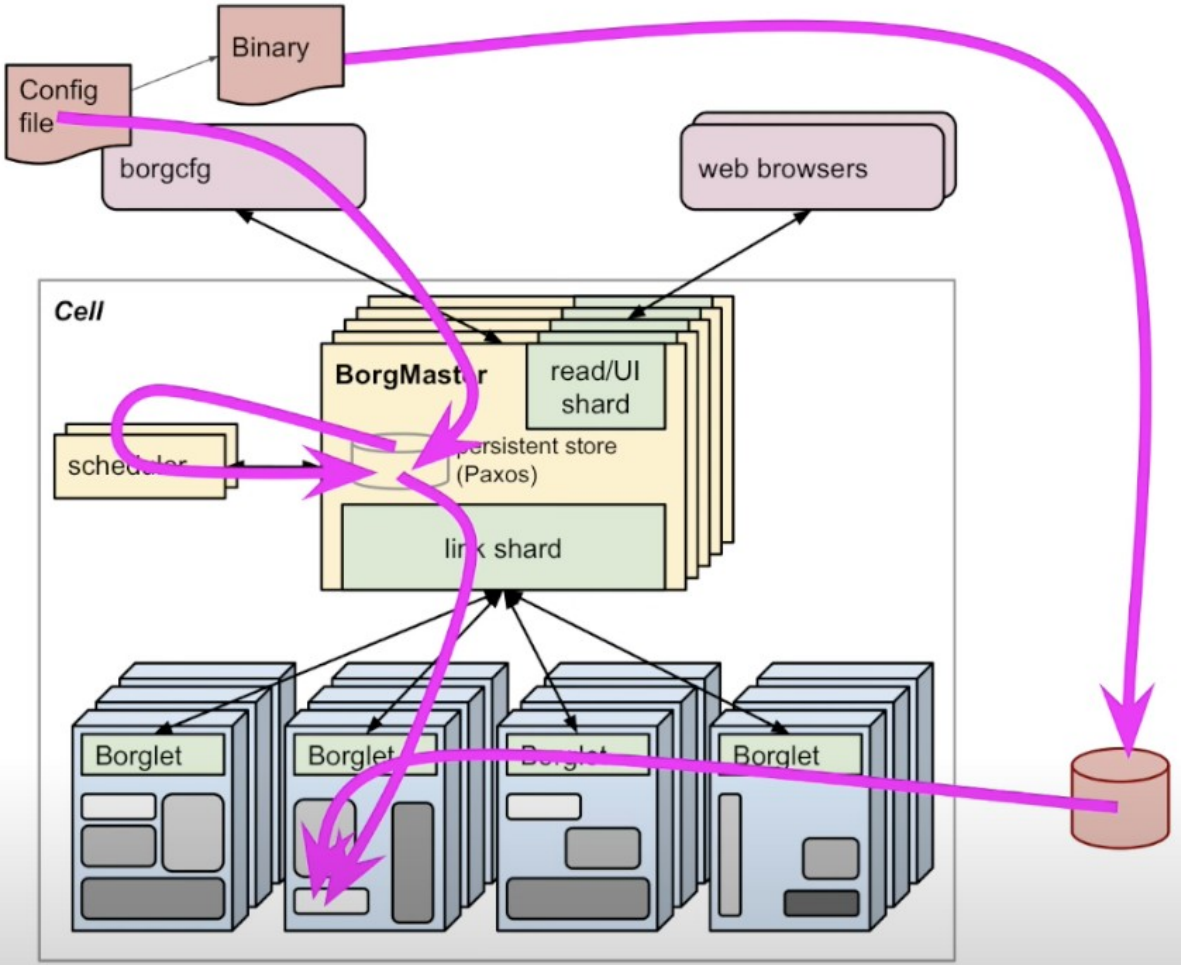
User view

```
job hello_world = {
   runtime = { cell = 'ic' }                  // Cell (cluster) to run in
   binary = '.../hello_world_webserver'       // Program to run
   args = { port = '%port%' }                 // Command line parameters
   requirements = {      // Resource requirements (optional)
     ram = 100M
     disk = 100M
     cpu = 0.1
   }
   replicas = 10000      // Number of tasks
}
```

10 years borg

User view

What just happened?



kubernetes Home Getting Started Community News Events Blog

Kubernetes

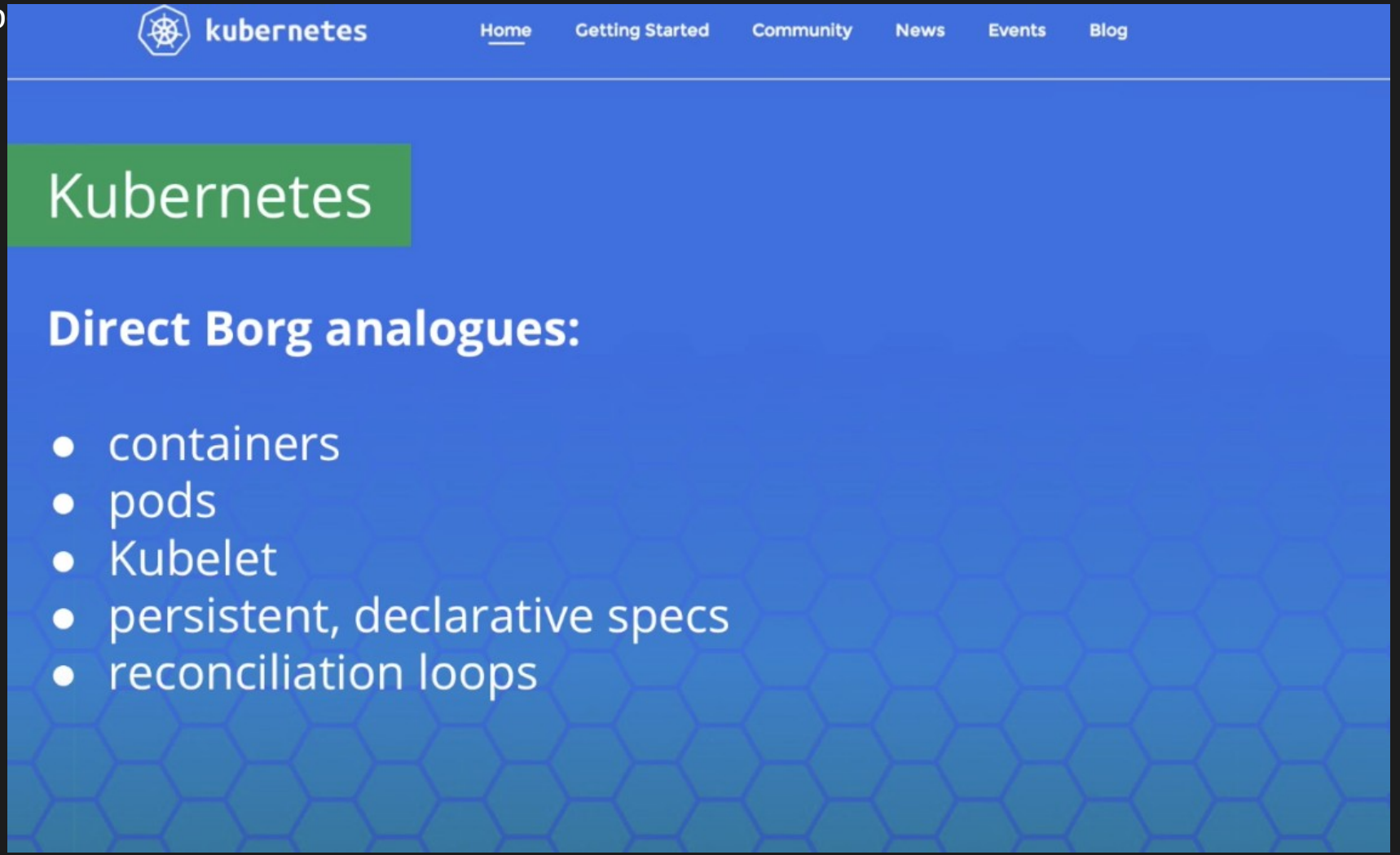
κυβερνήτης: *pilot or helmsman of a ship*

kubernetes by Google

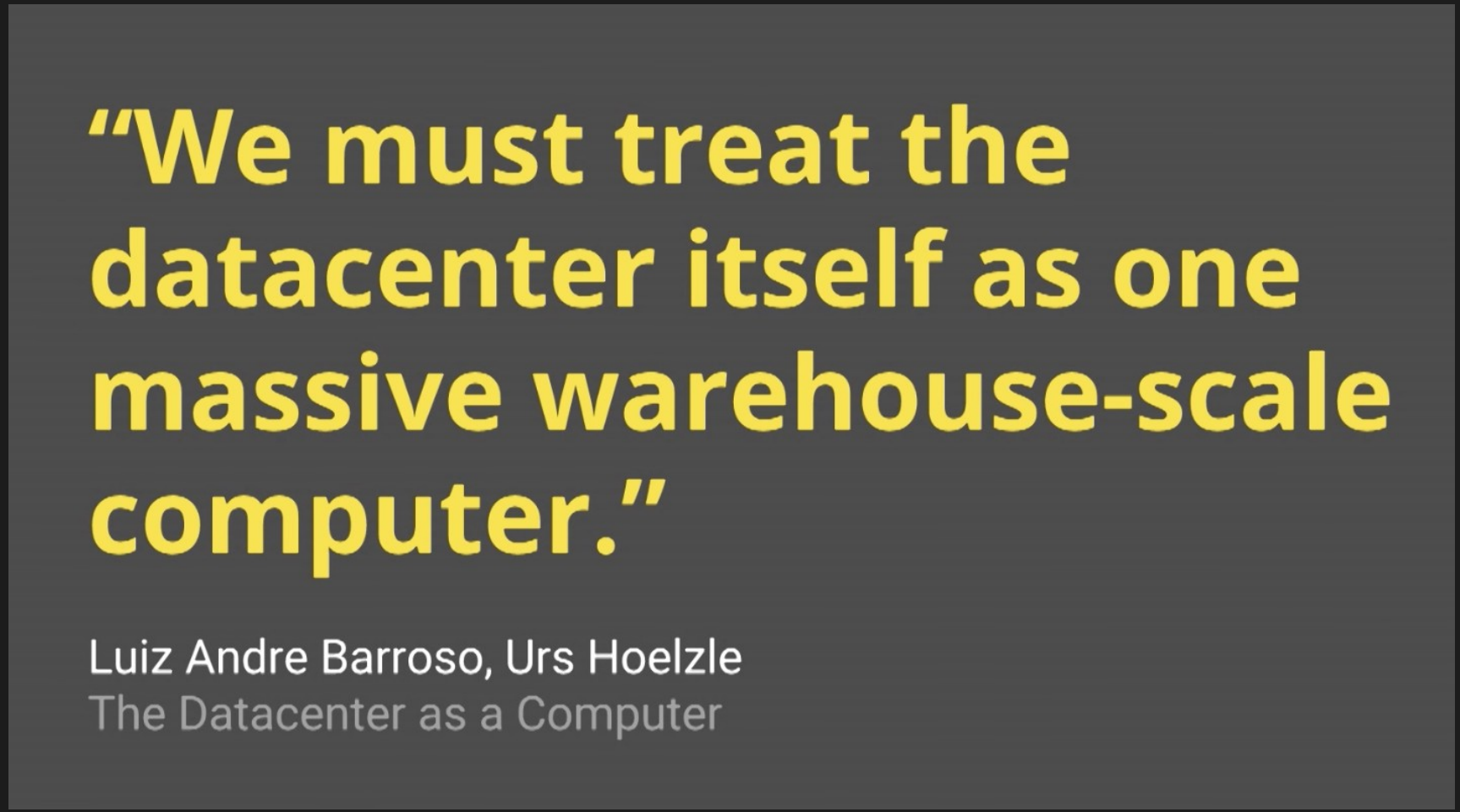
http://kubernetes.io

- Top 0.01% of all Github projects
- 800+ unique contributors
- 15000+ people signed up for k8s meetups

10 years b

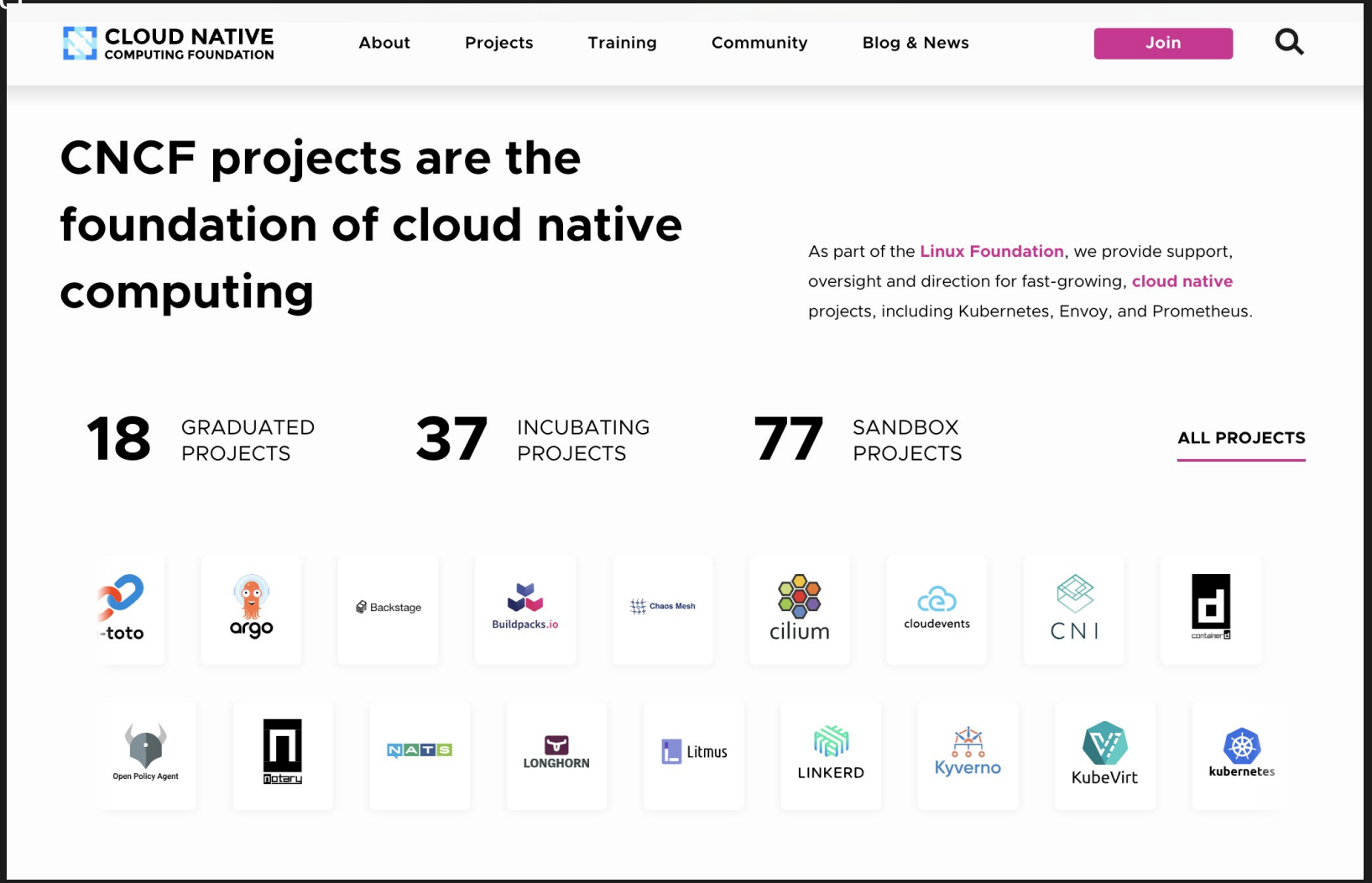

kubernetes
Home
Getting Started
Community
News
Events
Blog
Kubernetes
Direct Borg analogues:
containers
pods
Kubelet
persistent, declarative specs
reconciliation loops

One computer



One computer

One computer



pmi

The Art of War and Project Management.

Although it is not clear (Wikipedia, 2006) if Sun Tzu – Exhibit 1 - was one person or a group of philosophers, his pronouncements are extremely clear and definitely practical, no matter which Age is considering them.

Exhibit 1 – Sun Tzu?

Reading the Art of War I discovered strong collocations to Project Management.

The basis for all thoughts presented by Sun Tzu are five elements

described by Robert Friedler (2006) as: path, heaven, earth, leader and law (Exhibit 2). And should be the key things for any Project Manager to remember when managing a project.

	Direct translation	Meaning in the Art of War	Examples
	path	mutual philosophy & purpose of the group	patriotism, team sprit, shared values
	heaven	environmental factors	time, seasons, light, darkness
	earth	situation	distance from opponent, even or uneven ground
	leader	leadership	intelligence, credibility, compassion, courage, discipline
	law	art, as in "Art of War"	the ability to perceive and implement the strategic concept

Exhibit 2 - Sun Tzu's Five Elements

5 elements

	Direct translation	Meaning in the Art of War	Examples
	path	mutual philosophy & purpose of the group	patriotism, team sprit, shared values
	heaven	environmental factors	time, seasons, light, darkness
	earth	situation	distance from opponent, even or uneven ground
	leader	leadership	intelligence, credibility, compassion, courage, discipline
	law	art, as in "Art of War"	the ability to perceive and implement the strategic concept

Contents

k8s security

4c

k8s is not a security system

Sec wasn’t built in natively

k8s sec improving (1.22+)

K8s sec

OWASP Kubernetes Top Ten

Main

About the Kubernetes Top 10

When adopting Kubernetes, we introduce new risks to our applications and infrastructure. The The *OWASP Kubernetes Top 10* is aimed at helping security practitioners, system administrators, and software developers prioritize risks around the Kubernetes ecosystem. The Top Ten is a prioritized list of these risks backed by data collected from organizations varying in maturity and complexity.

- K00:2022 Welcome to the Kubernetes Security Top Ten
- K01:2022 Insecure Workload Configurations
- K02:2022 Supply Chain Vulnerabilities
- K03:2022 Overly Permissive RBAC Configurations
- K04:2022 Lack of Centralized Policy Enforcement
- K05:2022 Inadequate Logging and Monitoring
- K06:2022 Broken Authentication Mechanisms
- K07:2022 Missing Network Segmentation Controls
- K08:2022 Secrets Management Failures
- K09:2022 Misconfigured Cluster Components
- K10:2022 Outdated and Vulnerable Kubernetes Components
- Other Risks to Consider

K8s se

cloud native wiki
by aqua

- DevSecOps
- Docker Container
- Cloud Native Applications
- Kubernetes in Production
- Supply Chain Security
- Containerized Architecture
- Cloud Security
- Serverless Architecture
- Container Platforms
- Kubernetes
- Vulnerability Management

How Can You Best Secure Your Kubernetes (K8s) Deployment?

Kubernetes is a complex platform and requires extensive configuration and management. To keep Kubernetes workloads safe, especially in a production environment, you need to address key architectural vulnerabilities and platform dependencies, by implementing security best practices.

In this article, you will learn about the following Kubernetes security best practices:

Enable Role-Based Access Control (RBAC)

Use Third-Party Authentication for API Server

Protect ETCD with TLS and Firewall

Isolate Kubernetes Nodes

Monitor Network Traffic to Limit Communications

Use Process Whitelisting

Turn on Audit Logging

Keep Kubernetes Version Up to Date

Lock Down Kubelet

Secure Kubernetes with Aqua

K8s sec

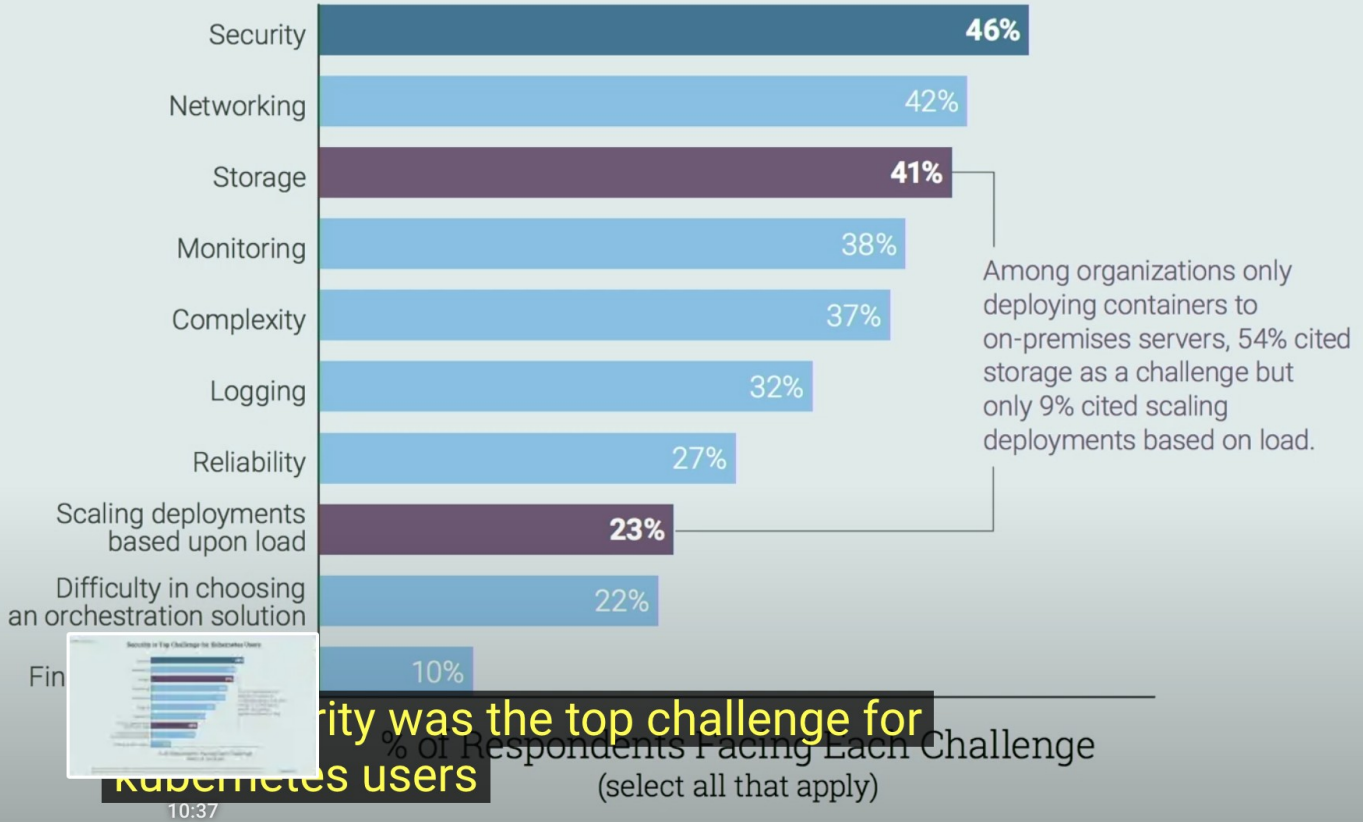
This guide describes the security challenges associated with setting up and securing a Kubernetes cluster. It includes strategies for system administrators and developers of National Security Systems, helping them avoid common misconfigurations and implement recommended hardening measures and mitigations when deploying Kubernetes. This guide details the following mitigations:

- Scan containers and Pods for vulnerabilities or misconfigurations.
- Run containers and Pods with the least privileges possible.
- Use network separation to control the amount of damage a compromise can cause.
- Use firewalls to limit unneeded network connectivity and use encryption to protect confidentiality.
- Use strong authentication and authorization to limit user and administrator access as well as to limit the attack surface.
- Capture and monitor audit logs so that administrators can be alerted to potential malicious activity.
- Periodically review all Kubernetes settings and use vulnerability scans to ensure risks are appropriately accounted for and security patches are applied.

K

@IanColdwater

Security is Top Challenge for Kubernetes Users



rity was the top challenge for Kubernetes users

10:37

Source: The New Stack Analysis of Cloud Native Computing Foundation survey conducted in Fall 2017. Q. What are your challenges in using/deploying containers? (check all that apply). n=527.Note, only respondents managing containers with Kubernetes were included in the ch

10:36 / 41:34

Ship of Fools: Shoring Up Kubernetes Security - SANS Secure DevOps Summit 2018

DIY Pen-Testing for Your

K8s se

juju.is/cloud-native-kubernetes-usage-report-2022

Juju Charmhub About Blog Learn Contribute

Key takeaways

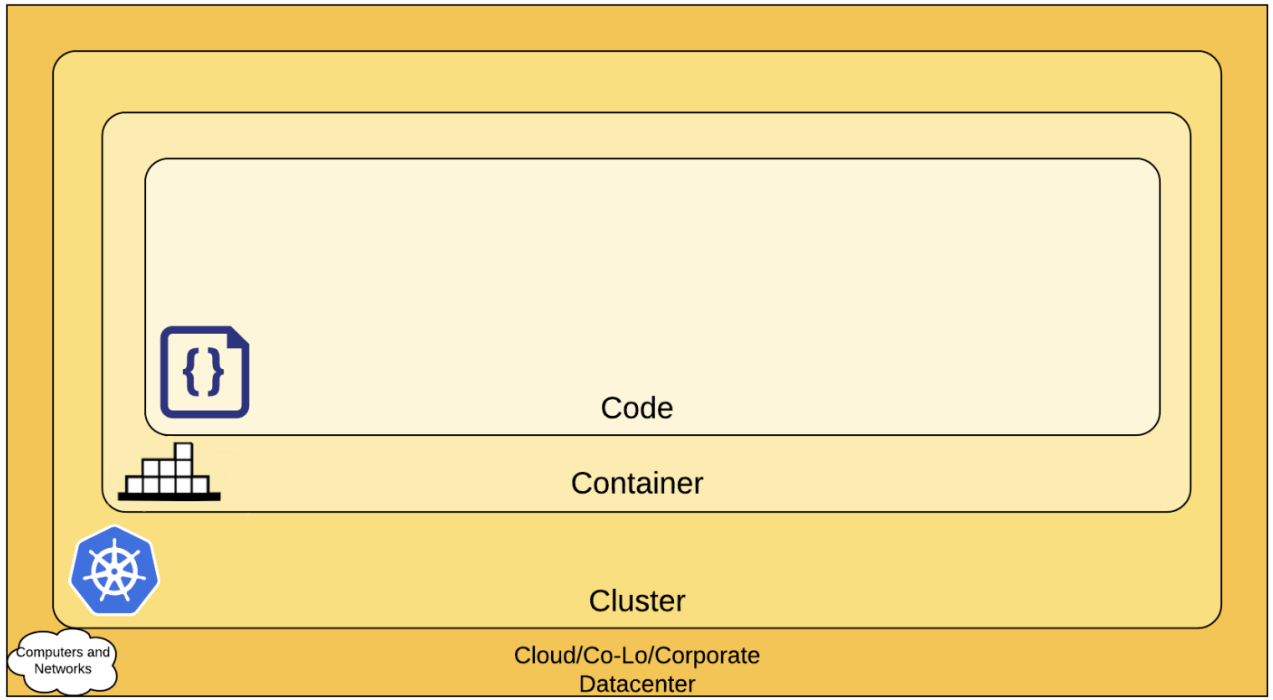
- **Hybrid vs multi-cloud: the reality behind the adoption**
  More than 83% of respondents told us they're using either hybrid or multi-cloud. In the last year alone, the percentage of respondents who did not use hybrid or multi-cloud dropped from 22.4% to 16.4%. This time, Tim Hockin discusses the reality behind that adoption: "People often build a straw man of hybrid or multi-cloud, with the idea of one giant mesh that spans the world and all the clouds, applications running wherever capacity is cheap and available. But in reality, that's not at all what people are doing with it. What they're actually doing is using each environment for just the things they have to use it for."
- **Kubernetes on bare metal**
  The question of where to run applications is an interesting one. 14% of respondents said that they run everything on Kubernetes, over 20% said on bare metal and VMs, and over 29% said a combination of bare metal, VMs, and Kubernetes. As highlighted by Ihor Dvoretskyi, this distribution shows how the flexibility of Kubernetes allows organisations to run the same type of workloads everywhere. Looking back at last year's highlight, where Kelsey Hightower stated that bare metal was a better choice for compute and resource-heavy use cases such as interactive machine learning jobs, it seems that the tune is changing. Actually, as running Kubernetes is becoming more accessible, Alexis Richardson speculates that organisations would further adopt Kubernetes on bare metal if they knew it was possible.
- **Security is still everyone's concern**
  This comes as no surprise: 38% of respondents suggest that security is the most important consideration whether when operating Kubernetes, building container images or defining an edge strategy. Keeping clusters up-to-date is a definitive best practice to solve security issues. However, according to Jose Miguel Parrella, it is not as embedded within the strategic-thinking IT infrastructure group as one could expect. Today, it is more of a Day-30 discussion that only occurs within the small team of Kubernetes maintainers of every organisation. Combined with the fact that only 13.5% of people reported that they've "mastered" security in the cloud native space, it is clear that organisations have some room to grow when it comes to properly adopting and managing Kubernetes in production.

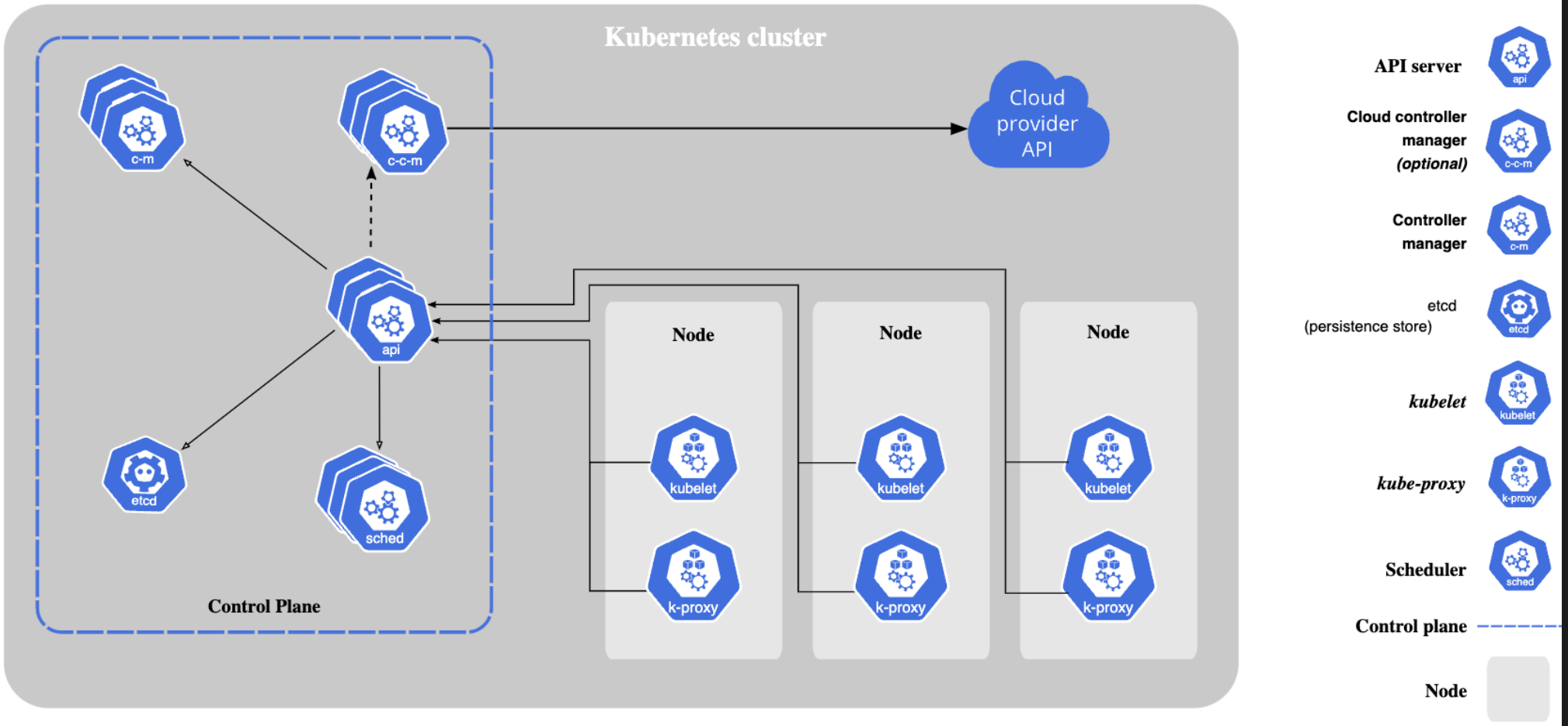
4c

The 4C's of Cloud Native security

You can think about security in layers. The 4C's of Cloud Native security are Cloud, Clusters, Containers, and Code.

> **Note:** This layered approach augments the defense in depth computing approach to security, which is widely regarded as a best practice for securing software systems.





The components of a Kubernetes cluster

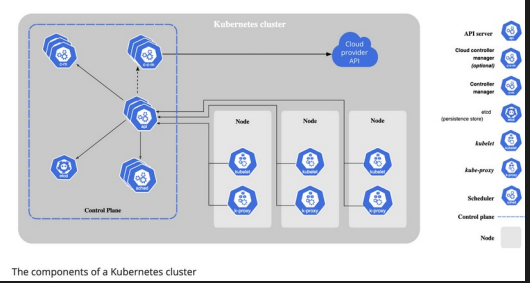
k8s control plane sec

cni install control plan
Apiserver
Kubelet
Schedulercontroler

Enable RBAC
Enable admission control (image pull policy, pod security standard)
Enable TLS
Remove anonymous

Network Policy
Secrets / ConfigMaps

etcd tls
etcd encryption

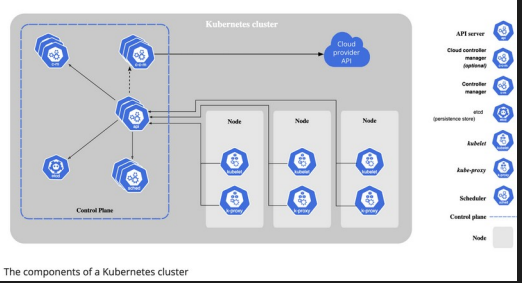


The components of a Kubernetes cluster

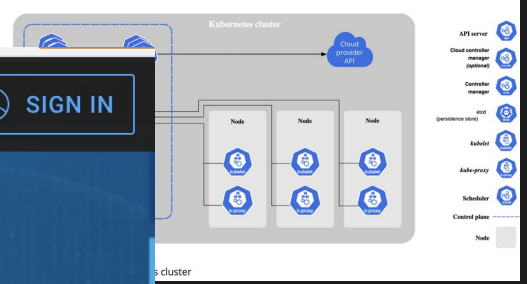
k8s node sec

Kubelet auth
Kubelet TLS
Kubelet config remove anonymous

Labels
Taints
Affinity
Resource Quotas

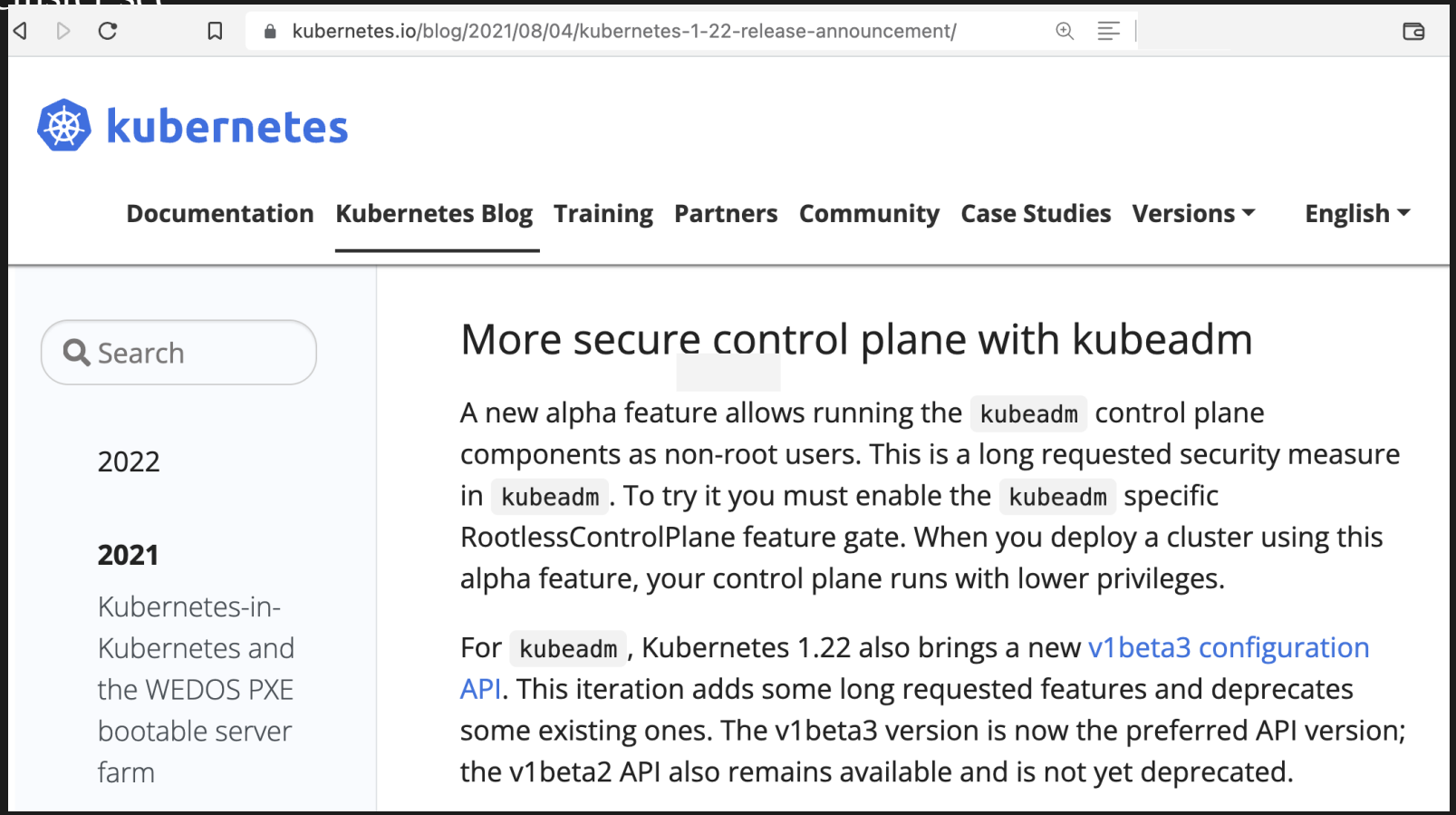


The components of a Kubernetes cluster

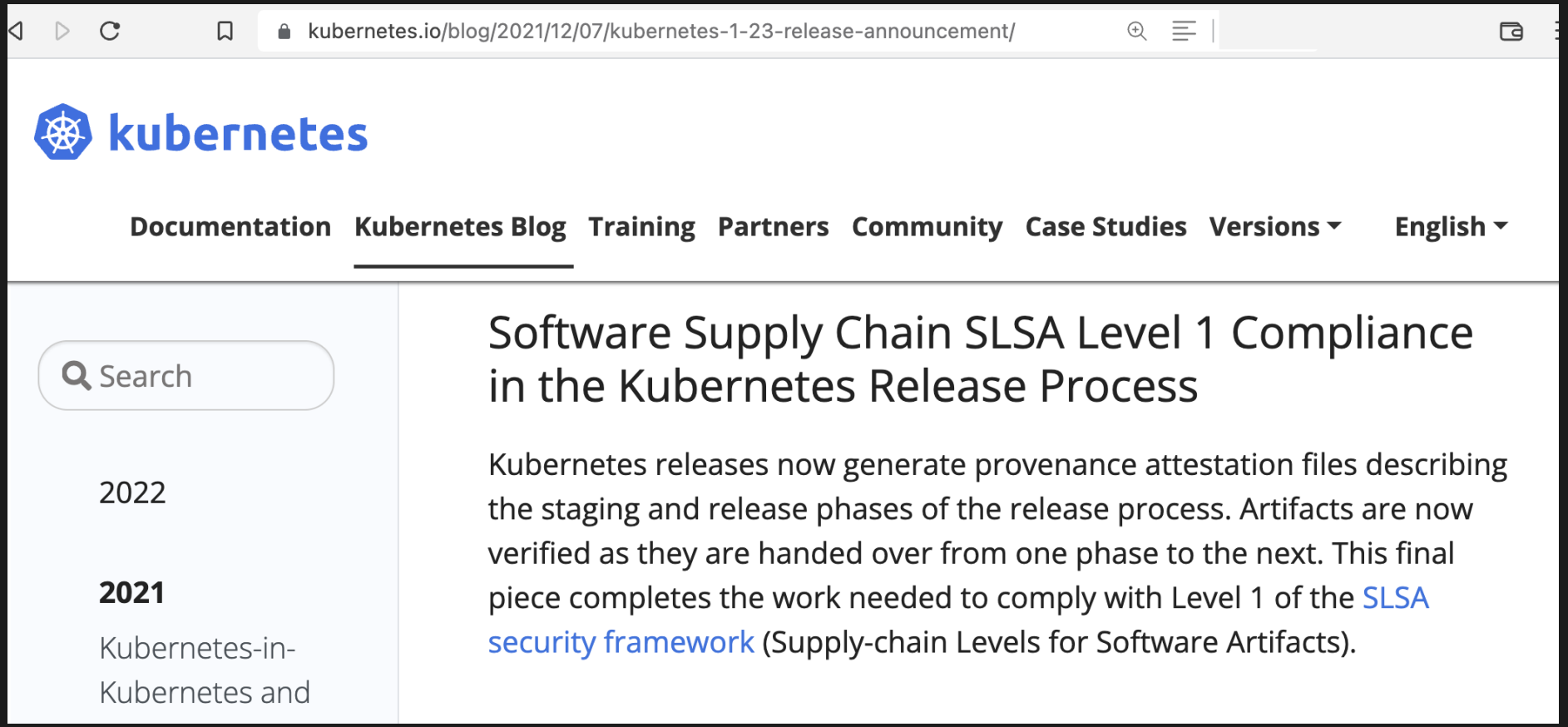


Kubernetes Attack and Defense: Break Out and Escalate!

k8s cluster sec

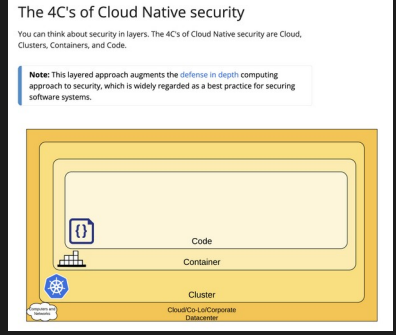


k8s cluster sec




kubernetes.io/blog/2022/05/03/kubernetes-1-24-release-announcement/
kubernetes
Documentation
Kubernetes Blog
Training
Partners
Community
Case Studies
Versions
English
Search
2022
Kubernetes 1.25: CSI Inline Volumes have graduated to
Signing Release Artifacts
Release artifacts are signed using cosign signatures, and there is experimental support for verifying image signatures. Signing and verification of release artifacts is part of increasing software supply chain security for the Kubernetes release process.

!k8s



code: languages (c++, java, python, js, rust, how about go?)
ci/cd, repo management, build / deploy / patch time
readability, compatibility -> attack surface

containers: base image, library, multi-stage, user, fs / v
image repo / scan, supply chain, deep dependency

data center: access control, server patch, configuration
redundancy, network

open source tools:
code: go simple, readability, promise backward compbality, static, compiled, grpc, no repo

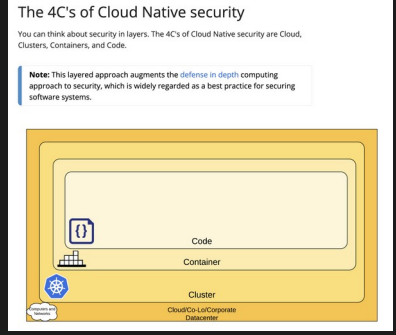
containers: Dockerfile, build tools, image repo, image scan trivy, kubebench, image pull polic

data center: server patch/upgrrade, apparmor/selinux, sysdig/falco

philosophical thinking

1 computer
1 network
1 tool

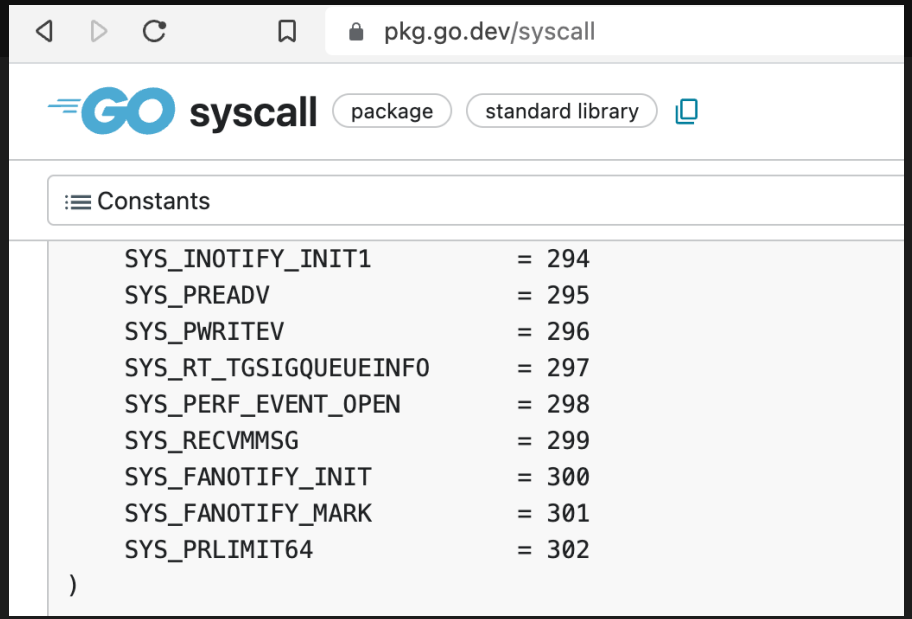
EBPF
XDP



ebpf

syscalls

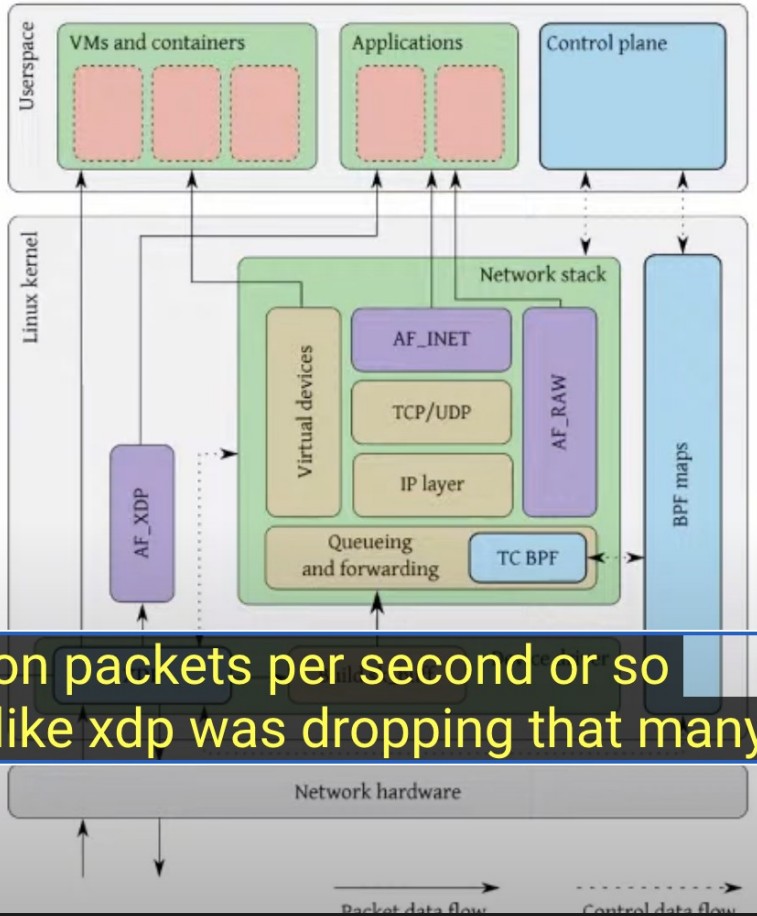
```
user@w02:/usr/src/linux-headers-5.15.0-46/include/linux$ grep "asmlinkage long sys" syscalls.h  | wc
    441    3020   27803
user@w02:/usr/src/linux-headers-5.15.0-46/include/linux$
user@w02:/usr/src/linux-headers-5.15.0-46/include/linux$ uname -r
5.15.0-46-generic
user@w02:/usr/src/linux-headers-5.15.0-46/include/linux$
user@w02:/usr/src/linux-headers-5.15.0-46/include/linux$ cat /etc/os-release | grep VERSION
VERSION_ID="22.04"
VERSION="22.04.1 LTS (Jammy Jellyfish)"
VERSION_CODENAME=jammy
user@w02:/usr/src/linux-headers-5.15.0-46/include/linux$
user@w02:/usr/src/linux-headers-5.15.0-46/include/linux$
```



xdp ddos drop pkts

g packet process-
h-cost approach,
ernal kernel APIs
is not surprising
se that have, the
4] virtual switch
ter frameworks,
a wide scope, al-
ety of uses. XDP
oving processing
nvironment, and
hus offering the
rface the kernel
can completely
erformance than
nto the existing

nto the operating
ys the programs
kets to a special
mal networking
to further lower
to the approach
RING [11], which



million packets per second or so
and like xdp was dropping that many

xdp ddos drop pkts

Droplet: DDoS Protection Framework

Programmability: abstract away interactions with user space

- Droplet handler: handles the dirty work
  - Runtime compilation
  - Kernel load/hook
- Different types of handlers
  - GenericHandler
  - IPHandler
  - PrefixHandler ...
- The user only needs to write BPF code in C

Contents

Demo
RBAC (ubuntu 3 nodes)
Network Policy (same)
eBPF/BPF (fedora / docker)
Cilium (minikube)

Contents

Demo
RBAC (ubuntu 3 nodes)

Contents

Demo
Network Policy (ubuntu 3 nodes)

network policy

```
user@cp01:~$ kubectl run web --image=nginx --labels="app=web" --expose --port=80
service/web created
pod/web created
user@cp01:~$ kubectl run --rm --labels="role=frontend-na"  -it --image=alpine test-$RANDOM  -- sh
If you don't see a command prompt, try pressing enter.
/ # wget -qO- http://web
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
/ #
```

```
user@cp01:~$ cat netpol-allow-labels.yaml
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: web-deny-all
spec:
  podSelector:
    matchLabels:
      app: web
  ingress:
    - from:
      - podSelector:
          matchLabels:
            role: frontend-na
user@cp01:~$
```

```
user@cp01:~$ kubectl run --rm --labels="role=frontend-xx"  -it --image=alpine test-$RANDOM  -- sh
If you don't see a command prompt, try pressing enter.
/ # wget -qO- --timeout=3 http://web
wget: download timed out
/ #
```

```
user@cp01:~$ cat netpol-allow-labels.yaml
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: web-deny-all
spec:
  podSelector:
    matchLabels:
      app: web
  ingress:
    - from:
      - podSelector:
          matchLabels:
            role: frontend-na
user@cp01:~$
```

Contents

Demo

eBPF/BPF (fedora / docker)

ebpf docker

```
pwd
/home/fedora/code/bpf_with_go

less Dockerfile

docker ps -a
docker rm 76b1
docker image ls
docker image rm 573

make
sudo docker run -it --privileged -v /sys/kernel/debug:/sys/kernel/debug "exec_scrape"
open another term, ssh, run cmds

#update cod
make
docker stop
sudo docker run -it --privileged -v /sys/kernel/debug:/sys/kernel/debug "exec_scrape"
```

ebpf docker

```
[fedora@fedora bpf_with_go]$ docker image ls
REPOSITORY                    TAG          IMAGE ID       CREATED         SIZE
exec_scrape                   latest       40eae72b44d1   9 seconds ago   3.52MB
<none>                        <none>       e0b0315859de   9 seconds ago   1.43GB
<none>                        <none>       cec2ba1929ea   3 minutes ago   1.43GB
<none>                        <none>       d47189e7c1f3   8 minutes ago   3.52MB
<none>                        <none>       69f22e335878   4 days ago      1.43GB
<none>                        <none>       010a5555d7f6   4 days ago      7.5GB
<none>                        <none>       e89228f2c12a   5 days ago      3.52MB
almalinux                     8.5          2e19f62a21cc   3 months ago    198MB
quay.io/cilium/ebpf-builder   1648566014   e10d558109c6   5 months ago    1.89GB
k8s.gcr.io/coredns/coredns    v1.8.6       a4ca41631cc7   10 months ago   46.8MB
k8s.gcr.io/pause              3.6          6270bb605e12   12 months ago   683kB
[fedora@fedora bpf_with_go]$ sudo docker run -it --privileged -v /sys/kernel/debug:/sys/kernel/debug "exec_scrap
On ***from devup cpu 03 containerd-shim ran : 3523 /usr/bin/runc
On ***from devup cpu 01 containerd-shim ran : 3529 /usr/bin/runc
On ***from devup cpu 02 bash ran : 3535 /usr/bin/clear
On ***from devup cpu 03 (sd-worker) ran : 3536 /usr/lib/systemd/systemd-userwo
On ***from devup cpu 03 (sd-worker) ran : 3537 /usr/lib/systemd/systemd-userwo
On ***from devup cpu 03 bash ran : 3539 /usr/bin/curl
On ***from devup cpu 03 bash ran : 3541 /usr/bin/git
On ***from devup cpu 01 bash ran : 3542 /usr/bin/su
On ***from devup cpu 03 bash ran : 3545 /usr/bin/su
```

ebpf docker

```
package main

//go:generate go run github.com/cilium/ebpf/cmd/bpf2go -target bpfel -cc clang gen_execve ./bpf/execve.bpf.c -- -I/usr/include/bpf -I.

import (
        "bytes"
        "encoding/binary"
        "fmt"
        "github.com/cilium/ebpf/link"
        "github.com/cilium/ebpf/perf"
        "golang.org/x/sys/unix"
        "log"
        "os"
)

type exec_data_t struct {
        Pid    uint32
        F_name [32]byte
"app/src/main.go" 67L, 1378B
```

ebpf c

```
func main() {
        setlimit()

        objs := gen_execveObjects{}

        loadGen_execveObjects(&objs, nil)
        link.Tracepoint("syscalls", "sys_enter_execve", objs.EnterExecve, nil)

        rd, err := perf.NewReader(objs.Events, os.Getpagesize())
        if err != nil {
                log.Fatalf("reader err")
        }

        for {
                ev, err := rd.Read()
                if err != nil {
                        log.Fatalf("Read fail")
                }

                if ev.LostSamples != 0 {
                        log.Printf("perf event ring buffer full, dropped %d samples", ev.LostSamples)
                        continue
                }

                b_arr := bytes.NewBuffer(ev.RawSample)

                var data exec_data_t
                if err := binary.Read(b_arr, binary.LittleEndian, &data); err != nil {
                        log.Printf("parsing perf event: %s", err)
                        continue
                }

                fmt.Printf("On ***from devup cpu %02d %s ran : %d %s\n",
                        ev.CPU, data.Comm, data.Pid, data.F_name)
```

Contents

Demo

Cilium (minikube)

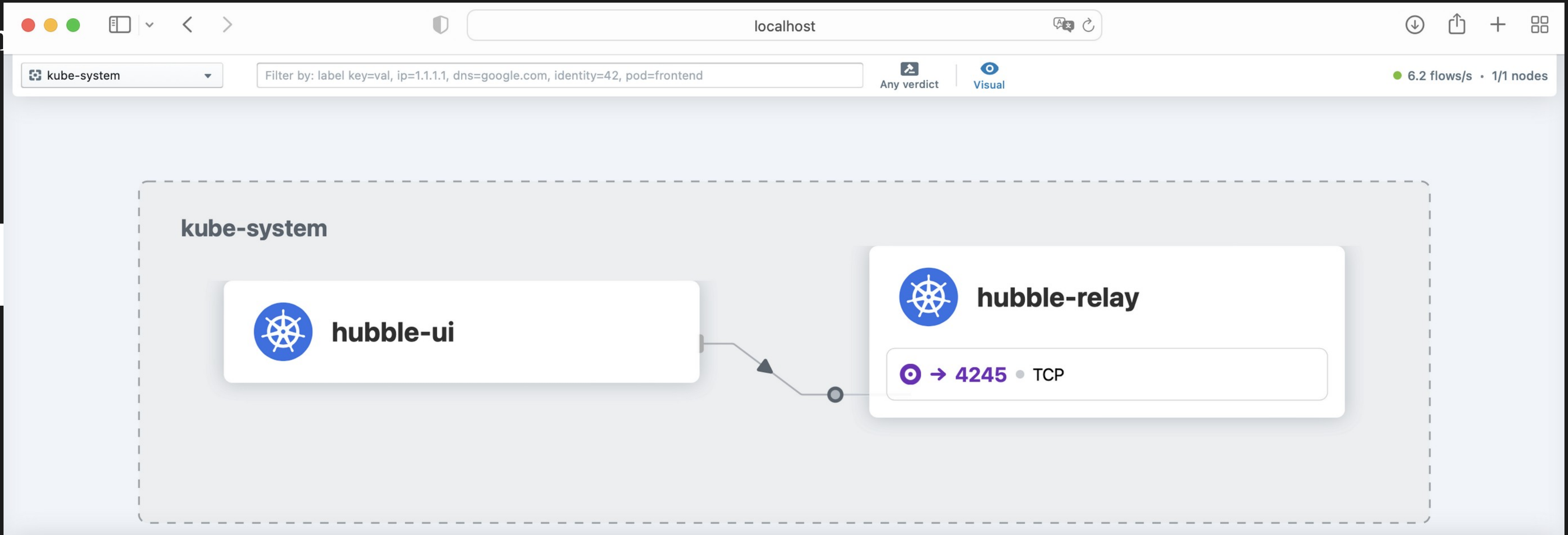
cilium

C

```
minikube version
# start
minikube start --network-plugin=cni --cni=fals --memory=4096
# install cilium
CILIUM_CLI_VERSION=$(curl -s https://raw.githubusercontent.com/cilium/cilium-cli/master/stable.txt)
CLI_ARCH=amd64
if [ "$(uname -m)" = "arm64" ]; then CLI_ARCH=arm64; fi
curl -L --fail --remote-name-all https://github.com/cilium/cilium-cli/releases/download/${CILIUM_CLI_VERSION}/cilium-darwin-${CLI_ARCH}.tar.gz{,.
shasum -a 256 -c cilium-darwin-${CLI_ARCH}.tar.gz.sha256sum
sudo tar xzvfC cilium-darwin-${CLI_ARCH}.tar.gz /usr/local/bin

# install
cilium install
cilium status
cilium hubble enable
cilium status

#install hubble client
export HUBBLE_VERSION=$(curl -s https://raw.githubusercontent.com/cilium/hubble/master/stable.txt)
HUBBLE_ARCH=amd64
if [ "$(uname -m)" = "arm64" ]; then HUBBLE_ARCH=arm64; fi
curl -L --fail --remote-name-all https://github.com/cilium/hubble/releases/download/$HUBBLE_VERSION/hubble-darwin-${HUBBLE_ARCH}.tar.gz{,.sha256s
shasum -a 256 -c hubble-darwin-${HUBBLE_ARCH}.tar.gz.sha256sum
sudo tar xzvfC hubble-darwin-${HUBBLE_ARCH}.tar.gz /usr/local/bin
#validate hubble api access
cilium hubble port-forward&
#
hubble status
hubble observe
# enable hubble ui
cilium hubble enable -ui
cilium hubble ui
```



Source Service	Destination Service	Destination Port	L7 info	Verdict	Timestamp
hubble-ui *kube-system*	hubble-relay *kube-system*	4245	—	forwarded	less than 5 seconds
hubble-ui *kube-system*	hubble-relay *kube-system*	4245	—	forwarded	less than 5 seconds
hubble-ui *kube-system*	hubble-relay *kube-system*	4245	—	forwarded	less than 5 seconds
hubble-ui *kube-system*	hubble-relay *kube-system*	4245	—	forwarded	less than 5 seconds
hubble-ui *kube-system*	hubble-relay *kube-system*	4245	—	forwarded	less than 5 seconds
hubble-ui *kube-system*	hubble-relay *kube-system*	4245	—	forwarded	less than 5 seconds
hubble-ui *kube-system*	hubble-relay *kube-system*	4245	—	forwarded	less than 5 seconds
hubble-ui *kube-system*	hubble-relay *kube-system*	4245	—	forwarded	less than 5 seconds
hubble-ui *kube-system*	hubble-relay *kube-system*	4245	—	forwarded	less than 10 seconds
hubble-ui *kube-system*	hubble-relay *kube-system*	4245	—	forwarded	less than 10 seconds
hubble-ui *kube-system*	hubble-relay *kube-system*	4245	—	forwarded	less than 10 seconds
hubble-ui *kube-system*	hubble-relay *kube-system*	4245	—	forwarded	less than 10 seconds
hubble-ui *kube-system*	hubble-relay *kube-system*	4245	—	forwarded	less than 10 seconds
hubble-ui *kube-system*	hubble-relay *kube-system*	4245	—	forwarded	less than 10 seconds
hubble-ui *kube-system*	hubble-relay *kube-system*	4245	—	forwarded	less than 10 seconds
hubble-ui *kube-system*	hubble-relay *kube-system*	4245	—	forwarded	less than 10 seconds
hubble-ui *kube-system*	hubble-relay *kube-system*	4245	—	forwarded	less than 20 seconds
hubble-ui *kube-system*	hubble-relay *kube-system*	4245	—	forwarded	less than 20 seconds
hubble-ui *kube-system*	hubble-relay *kube-system*	4245	—	forwarded	less than 20 seconds
hubble-ui *kube-system*	hubble-relay *kube-system*	4245	—	forwarded	less than 20 seconds
hubble-ui *kube-system*	hubble-relay *kube-system*	4245	—	forwarded	less than 20 seconds
hubble-ui *kube-system*	hubble-relay *kube-system*	4245	—	forwarded	less than 20 seconds

techbeacon.com/security/tesla-drives-cryptojack-gangs-aws-cloud-down-kubernetes-avenue

App Dev & Testing | Enterprise IT | Security | GUIDES | CONFERENCES | SUBSCRIBE

Home / Security / Information Security

Security Blogwatch

Tesla drives cryptojack gang's AWS cloud down Kubernetes avenue

Richi Jennings
Your humble blogwatcher, dba RJA

A Tesla-owned AWS account was hacked to mine Monero. Who knows when; who knows for how long. Shocking.

The hackers drove straight in using an "unsecured" Kubernetes admin console (i.e., it had no password). They ran the *WannaMine* fileless APT on auto-pilot—and might also have stolen some sensitive data from Tesla drivers.

It's an electrifying story. In this week's *Security Blogwatch*, we're charged with plugging in the route of the tale, and bringing you up to speed.

Accessible URL

Admin Roles

cuckoo’s game

Hardback copies of my first book, *The Cuckoo's Egg*. Hardback. New old stock, unread. 323 pages printed on the finest quality 55 pound Glatfelter paper. *ALL GONE* *The last one left here on July 13, 2014 -- and my wife is delighted to reclaim our closet!*

This book is straight from 1989, back before my hair turned gray. It's the true story about how I caught a spy over the early Internet.. Some of the technology feels antique (2400 baud modems), and the author feels a quarter century older. So get a copy while I'm still able to sign it..

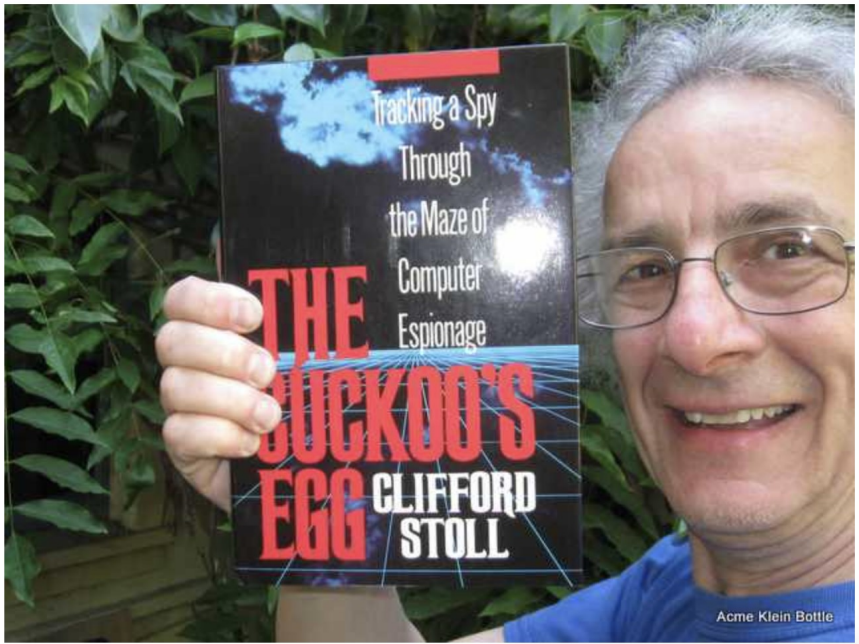
By default, I will autograph this book to you. Alternatively, I'm happy to sign it to a friend of yours, or leave it unsigned. (tell me what to do in the comments section of the order form)

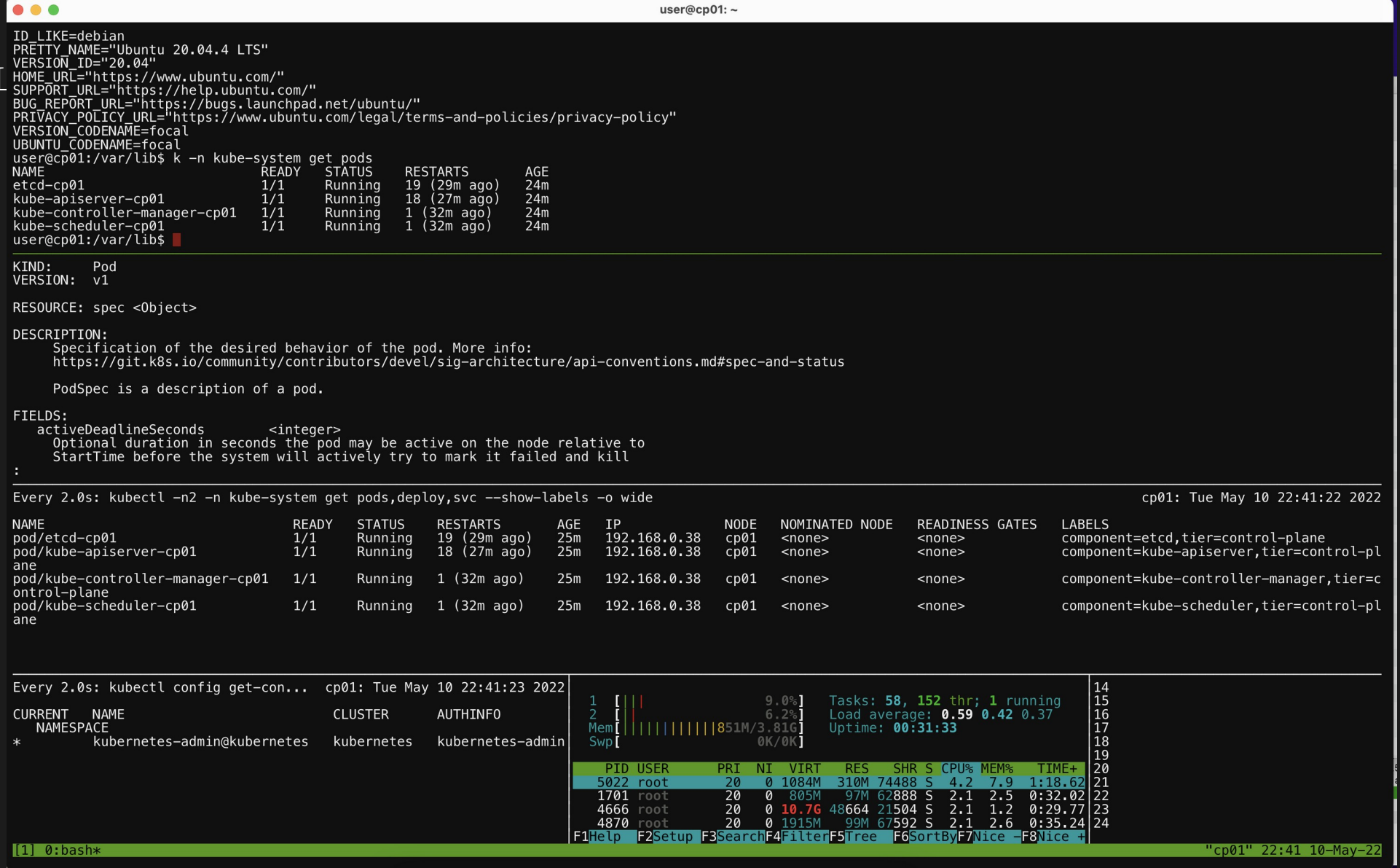
Send me email if you want a copy of Cuckoo's Egg in another language - I have a few in Czech and Dutch; possibly one or two in Spanish and Chinese, and maybe a couple other languages.

Each of these books comes complete with book jacket, 323 pages, lots of nouns, verbs, prepositions, and even a few participles (whatever thosea are). I guarantee that I wrote the whole book (six months of pecking on a MacPlus during 1988)

If you want 10 copies, oh, do I have a deal for you! (yep, I have about two dozen of these books, still in their original boxes) *SOLD - GONE - NO MORE*

Buy a copy of The Cuckoo's Egg and make my wife really happy...




user@cp01: ~
ID_LIKE=debian
PRETTY_NAME="Ubuntu 20.04.4 LTS"
VERSION_ID="20.04"
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
VERSION_CODENAME=focal
UBUNTU_CODENAME=focal
user@cp01:/var/lib$ k -n kube-system get pods
NAME READY STATUS RESTARTS AGE
etcd-cp01 1/1 Running 19 (29m ago) 24m
kube-apiserver-cp01 1/1 Running 18 (27m ago) 24m
kube-controller-manager-cp01 1/1 Running 1 (32m ago) 24m
kube-scheduler-cp01 1/1 Running 1 (32m ago) 24m
user@cp01:/var/lib$
KIND: Pod
VERSION: v1
RESOURCE: spec <Object>
DESCRIPTION:
Specification of the desired behavior of the pod. More info:
https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status
PodSpec is a description of a pod.
FIELDS:
activeDeadlineSeconds <integer>
Optional duration in seconds the pod may be active on the node relative to
StartTime before the system will actively try to mark it failed and kill
:
Every 2.0s: kubectl -n2 -n kube-system get pods,deploy,svc --show-labels -o wide cp01: Tue May 10 22:41:22 2022
NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES LABELS
pod/etcd-cp01 1/1 Running 19 (29m ago) 25m 192.168.0.38 cp01 <none> <none> component=etcd,tier=control-plane
pod/kube-apiserver-cp01 1/1 Running 18 (27m ago) 25m 192.168.0.38 cp01 <none> <none> component=kube-apiserver,tier=control-pl
ane
pod/kube-controller-manager-cp01 1/1 Running 1 (32m ago) 25m 192.168.0.38 cp01 <none> <none> component=kube-controller-manager,tier=c
ontrol-plane
pod/kube-scheduler-cp01 1/1 Running 1 (32m ago) 25m 192.168.0.38 cp01 <none> <none> component=kube-scheduler,tier=control-pl
ane
Every 2.0s: kubectl config get-con... cp01: Tue May 10 22:41:23 2022
CURRENT NAME CLUSTER AUTHINFO
NAMESPACE
* kubernetes-admin@kubernetes kubernetes kubernetes-admin
1 [||| 9.0%] Tasks: 58, 152 thr; 1 running
2 [|| 6.2%] Load average: 0.59 0.42 0.37
Mem[||||||||||||851M/3.81G] Uptime: 00:31:33
Swp[0K/0K]
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+
5022 root 20 0 1084M 310M 74488 S 4.2 7.9 1:18.62
1701 root 20 0 805M 97M 62888 S 2.1 2.5 0:32.02
4666 root 20 0 10.7G 48664 21504 S 2.1 1.2 0:29.77
4870 root 20 0 1915M 99M 67592 S 2.1 2.6 0:35.24
F1Help F2Setup F3SearchF4FilterF5Tree F6SortByF7Nice -F8Nice +
14
15
16
17
18
19
20
21
22
23
24
[1] 0:bash*
"cp01" 22:41 10-May-22

Art of wars

Know sylf know thee, hund battles hund wins.
Know sylf not thee, half win.
Know not sylf nor thee, lose all.

Winning without fighting

Nature
Opportunity
Environment
Leadership
Tools

sources learning

Public domains
Open source codes/tools

```
func DoNext(b ...labs, t ...tools, i ...ideas) {
    defer SeeUsAfter()
    go ThankDevUPMgrGTL(){ <-BetterJobs }()
}
```

end

$$S = f(K, S, T)^{p}$$

Success k8s sec = f (Knowledge, Skills, Tools) power of people