
KubeDay
JAPAN
JAPAN

KubeDay
JAPAN

Secure and Debuggable: Debugging Slim, Scratch and Distroless Kubernetes Containers

Saiyam Pathak & Kyle Quest

Intro

- **Expected** background
  - basic container fundamentals
  - basic knowledge of namespaces
  - k8s fundamental
- **What** will be covered
- **Who** we are

About me

- Director of Technical Evangelism, Civo
- CNCF Ambassador
- CKA, CKAD, CKS Certified
- KCNA Certification SME

saiyampathak.com/twitter

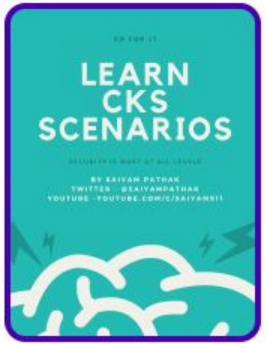
saiyampathak.com/youtube

saiyampathak.com/discord

kubesimplify.com

Let's Learn CKS Scenarios

Author of a book which helps you prepare for the Kubernetes CKS exam



Kyle Quest

(aka the docker-slim guy)

- DockerSlim creator
- Go enthusiast
  - 50 Shades of Go
  - Go Project Layout
- eBPF hacker
- Slim.AI founder/CTO
- https://twitter.com/kcqon
- https://github.com/kcq

The Problem

“

This minimal container image worked fine, but only if I never made a mistake.

General Debugging Techniques for Kubernetes

- Finding/tracking failures in kubernetes (events, logs, kubectl commands)
- Host/node level debugging
  - traditional (ssh-based)
  - using privileged containers and nsenter
- Embedding debugging tools
- Copying debugging tools
- Interactive container debugging with kubectl exec
- Permanent observability/debugging sidecars

Troubleshoting containers

```
STATUS              RESTARTS
Pending             0
Running             0
CrashLoopBackOff    1 (7s ago)
```

crashLoopBackOff

kubectl describe → See events
Pending pod → Insufficient resources

kubectl logs pod --previous

kubetl exec -it pod -- sh/bash → For running containers with debugging utilities

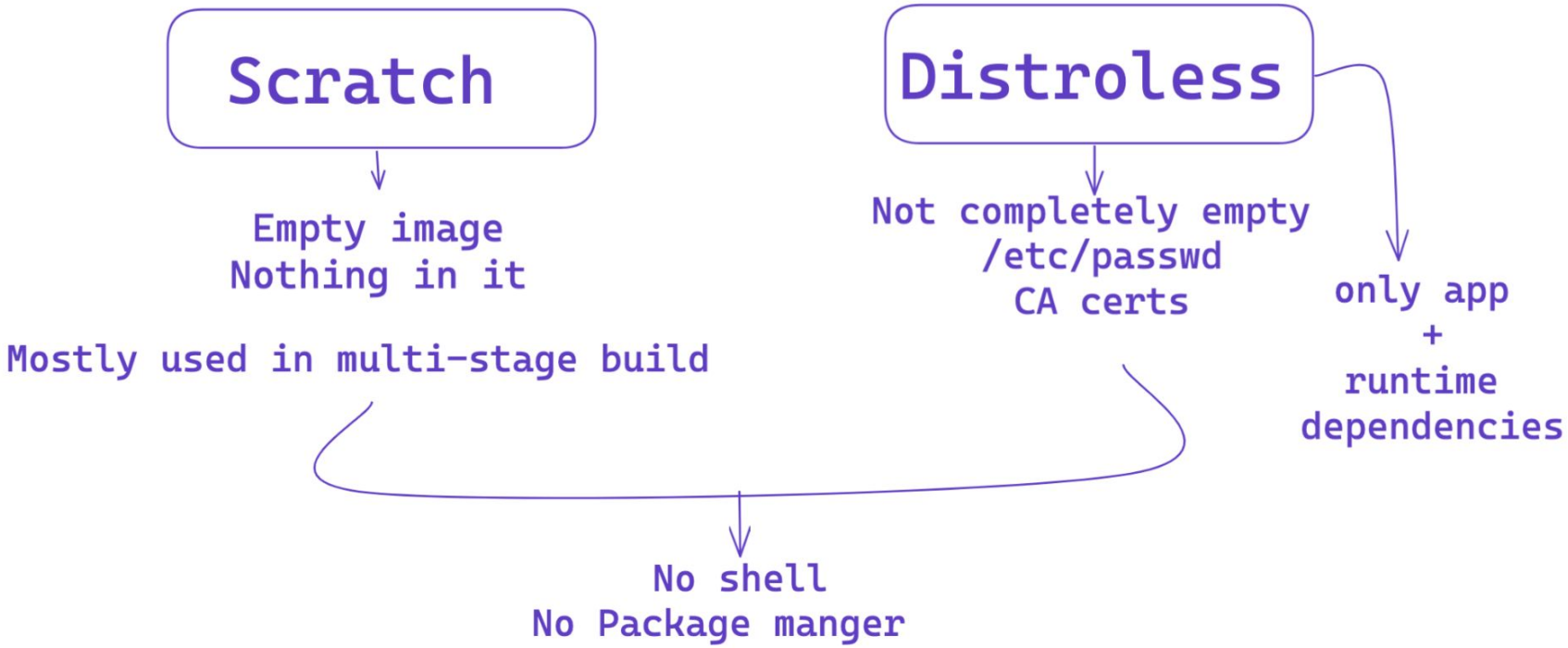
Kubectl get events

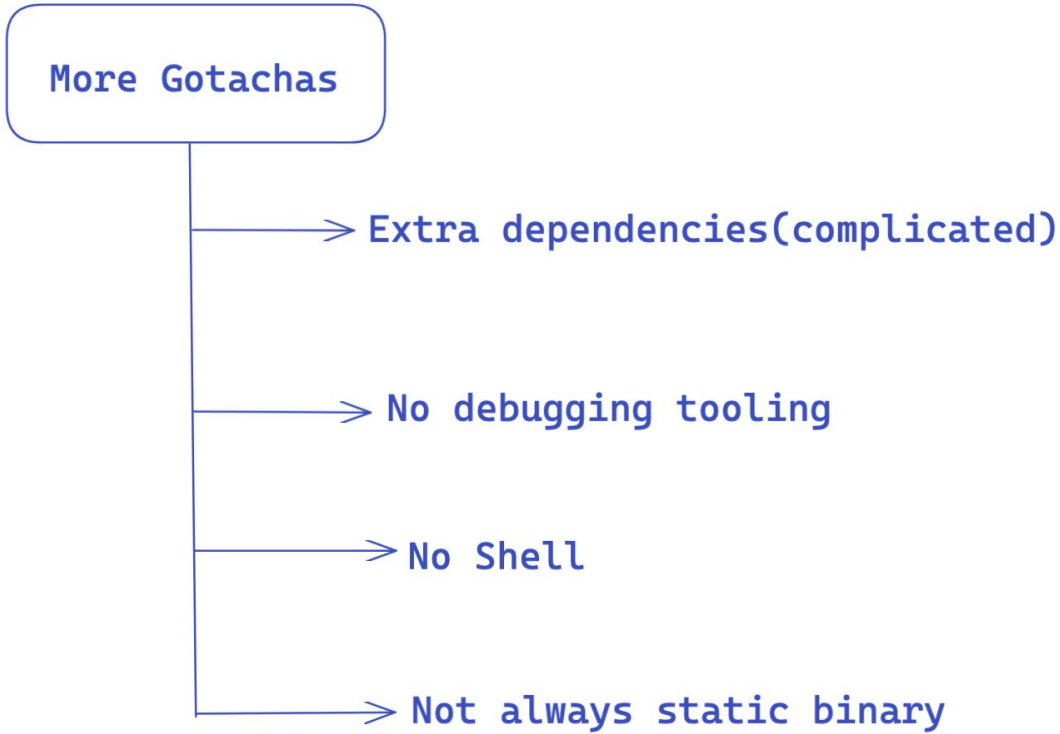
```
Events:
  Type     Reason            Age   From               Message
  ----     ------            ----  ----               -------
  Warning  FailedScheduling  21s   default-scheduler  0/3 nodes a
re available: 3 node(s) didn't match Pod's node affinity/selector
```

```
demo ~ kubectl exec -it nginx -- sh
# ls
bin   docker-entrypoint.d   home   media  proc  sbin  tmp
boot  docker-entrypoint.sh  lib    mnt    root  srv   usr
dev   etc                   lib64  opt    run   sys   var
```

Slim and Minimal Container Image

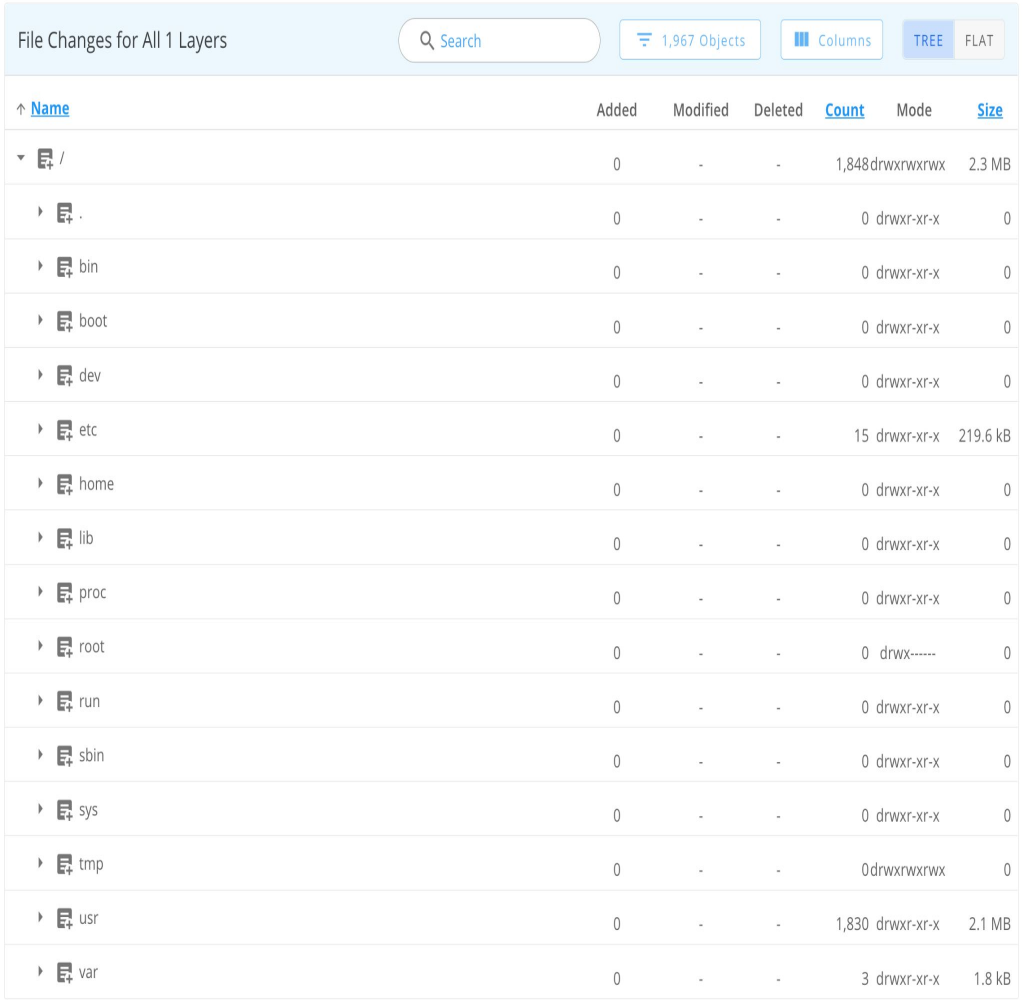
- **Why**
- **How**/options
  - Scratch
  - Distroless
  - Slim
- **Gotchas**


Scratch
Empty image
Nothing in it
Mostly used in multi-stage build
Distroless
Not completely empty
/etc/passwd
CA certs
only app
+
runtime
dependencies
No shell
No Package manger


More Gotachas
Extra dependencies(complicated)
No debugging tooling
No Shell
Not always static binary

Creating Minimal Container Images

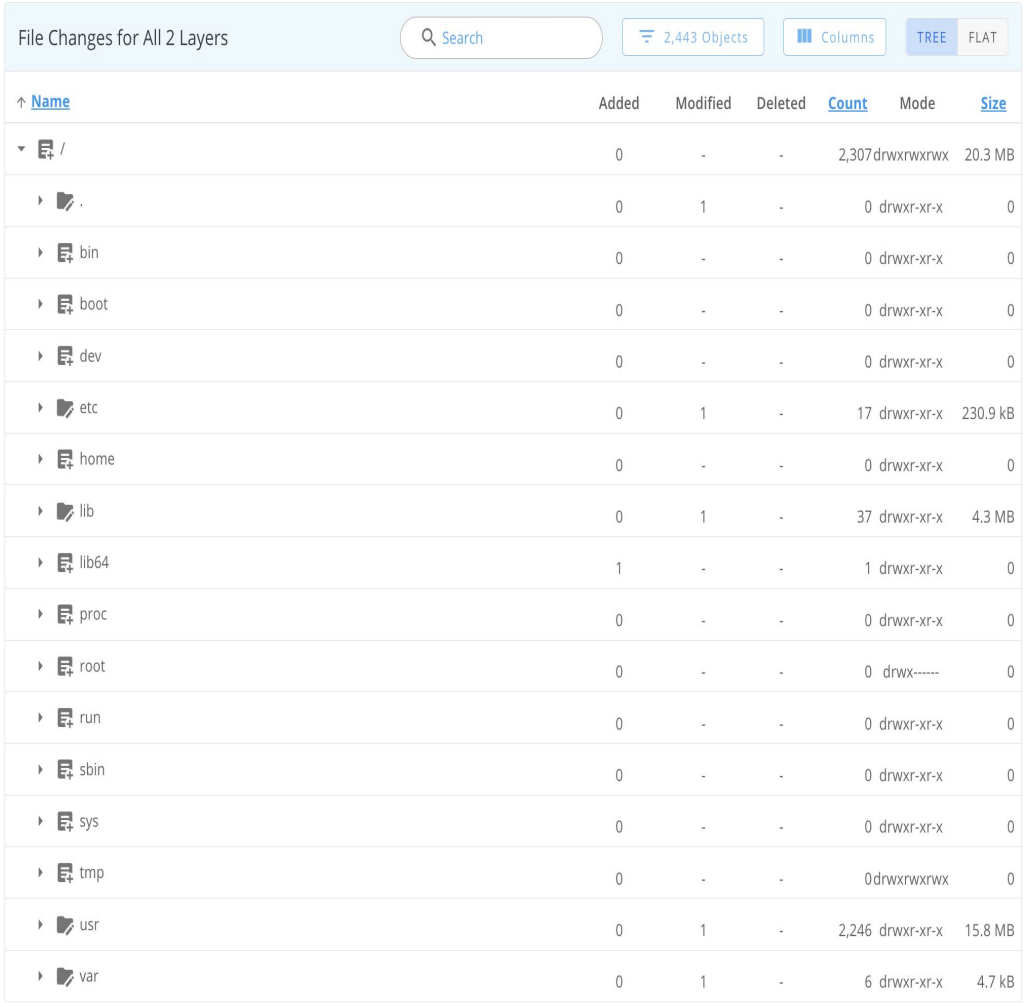
```
% nerdctl -n k8s.io images

REPOSITORY                                TAG                 IMAGE ID        SIZE

ghcr.io/kubeday-japan/demo-node-app fat                 13db159790c7      1.0 GiB
ghcr.io/kubeday-japan/demo-node-app distroless          b9ce468dce45    166.8 MiB
ghcr.io/kubeday-japan/demo-node-app slim                412eeb9bdb61     90.9 MiB
```



Name	Added	Modified	Deleted	Count	Mode	Size
/	0	-	-	1,848	drwxrwxrwx	2.3 MB
.	0	-	-	0	drwxr-xr-x	0
bin	0	-	-	0	drwxr-xr-x	0
boot	0	-	-	0	drwxr-xr-x	0
dev	0	-	-	0	drwxr-xr-x	0
etc	0	-	-	15	drwxr-xr-x	219.6 kB
home	0	-	-	0	drwxr-xr-x	0
lib	0	-	-	0	drwxr-xr-x	0
proc	0	-	-	0	drwxr-xr-x	0
root	0	-	-	0	drwx------	0
run	0	-	-	0	drwxr-xr-x	0
sbin	0	-	-	0	drwxr-xr-x	0
sys	0	-	-	0	drwxr-xr-x	0
tmp	0	-	-	0	drwxrwxrwx	0
usr	0	-	-	1,830	drwxr-xr-x	2.1 MB
var	0	-	-	3	drwxr-xr-x	1.8 kB

distroless/static - 2.34MB

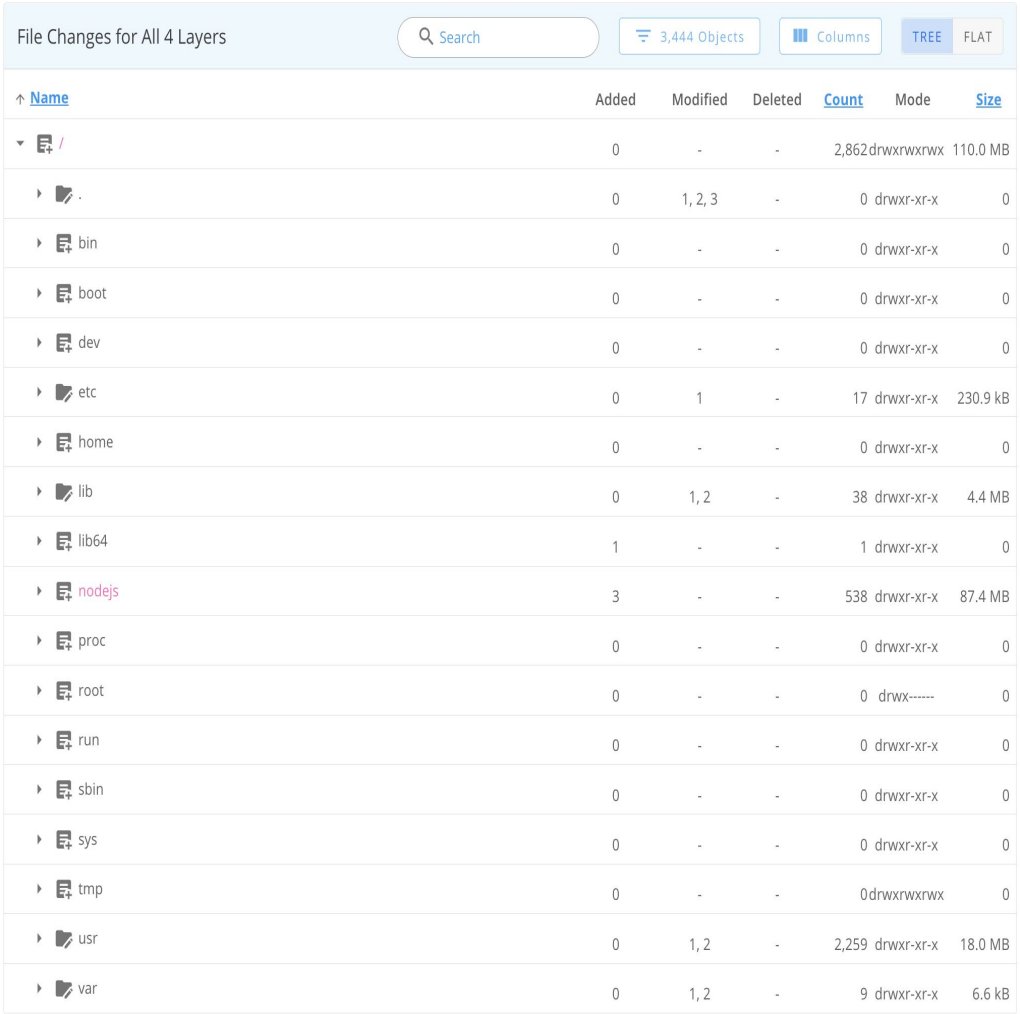
- Basic FS layout
- **/usr/share/zoneinfo** - 1.7 MB



Name	Added	Modified	Deleted	Count	Mode	Size
/	0	-	-	2,307	drwxrwxrwx	20.3 MB
.	0	1	-	0	drwxr-xr-x	0
bin	0	-	-	0	drwxr-xr-x	0
boot	0	-	-	0	drwxr-xr-x	0
dev	0	-	-	0	drwxr-xr-x	0
etc	0	1	-	17	drwxr-xr-x	230.9 kB
home	0	-	-	0	drwxr-xr-x	0
lib	0	1	-	37	drwxr-xr-x	4.3 MB
lib64	1	-	-	1	drwxr-xr-x	0
proc	0	-	-	0	drwxr-xr-x	0
root	0	-	-	0	drwx------	0
run	0	-	-	0	drwxr-xr-x	0
sbin	0	-	-	0	drwxr-xr-x	0
sys	0	-	-	0	drwxr-xr-x	0
tmp	0	-	-	0	drwxrwxrwx	0
usr	0	1	-	2,246	drwxr-xr-x	15.8 MB
var	0	1	-	6	drwxr-xr-x	4.7 kB

distroless/base - 20.3MB

- distroless/static++
- **/lib/x86_64-linux-gnu/** - 4.3MB
- **/usr/lib/x86_64-linux-gnu/** - 11.7MB



Name	Added	Modified	Deleted	Count	Mode	Size
/	0	-	-	2,862	drwxrwxrwx	110.0 MB
.	0	1, 2, 3	-	0	drwxr-xr-x	0
bin	0	-	-	0	drwxr-xr-x	0
boot	0	-	-	0	drwxr-xr-x	0
dev	0	-	-	0	drwxr-xr-x	0
etc	0	1	-	17	drwxr-xr-x	230.9 kB
home	0	-	-	0	drwxr-xr-x	0
lib	0	1, 2	-	38	drwxr-xr-x	4.4 MB
lib64	1	-	-	1	drwxr-xr-x	0
nodejs	3	-	-	538	drwxr-xr-x	87.4 MB
proc	0	-	-	0	drwxr-xr-x	0
root	0	-	-	0	drwx------	0
run	0	-	-	0	drwxr-xr-x	0
sbin	0	-	-	0	drwxr-xr-x	0
sys	0	-	-	0	drwxr-xr-x	0
tmp	0	-	-	0	drwxrwxrwx	0
usr	0	1, 2	-	2,259	drwxr-xr-x	18.0 MB
var	0	1, 2	-	9	drwxr-xr-x	6.6 kB

distroless/nodejs - 110MB

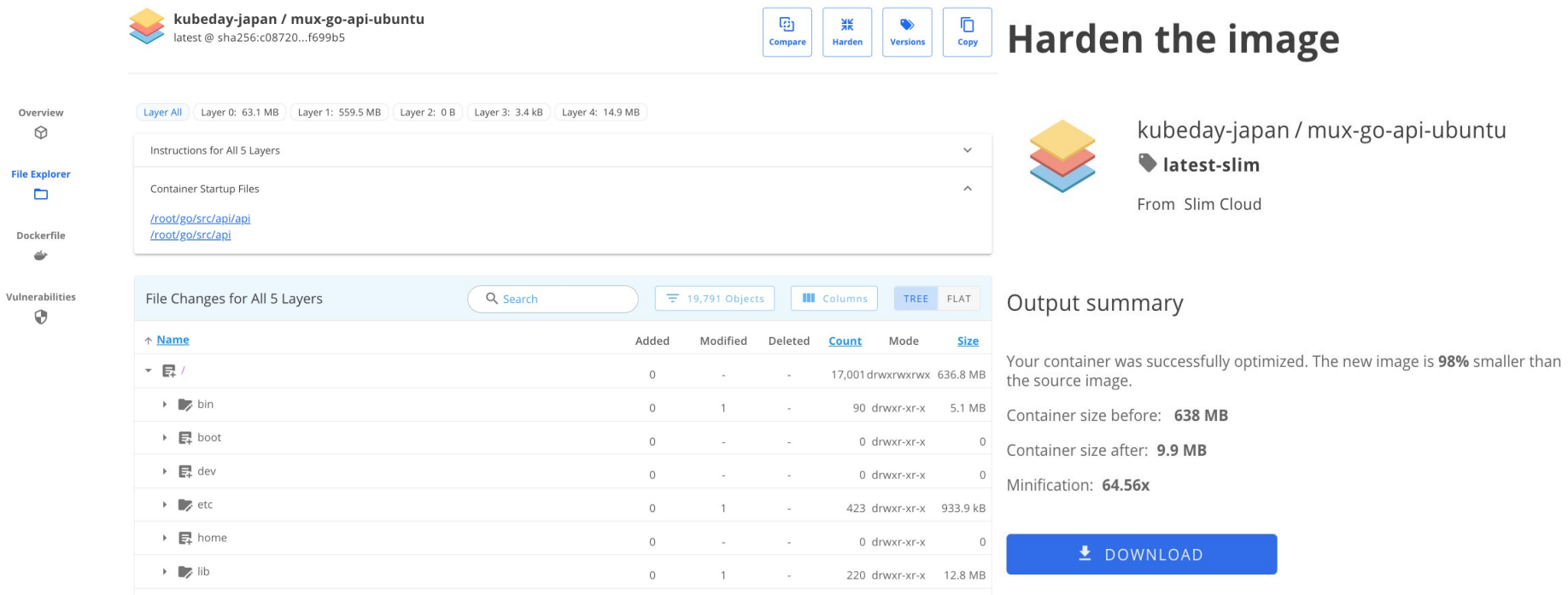
- distroless/base++
- **/usr/lib/x86_64-linux-gnu/** - 13.9MB
- **/nodejs/** - 87.4MB

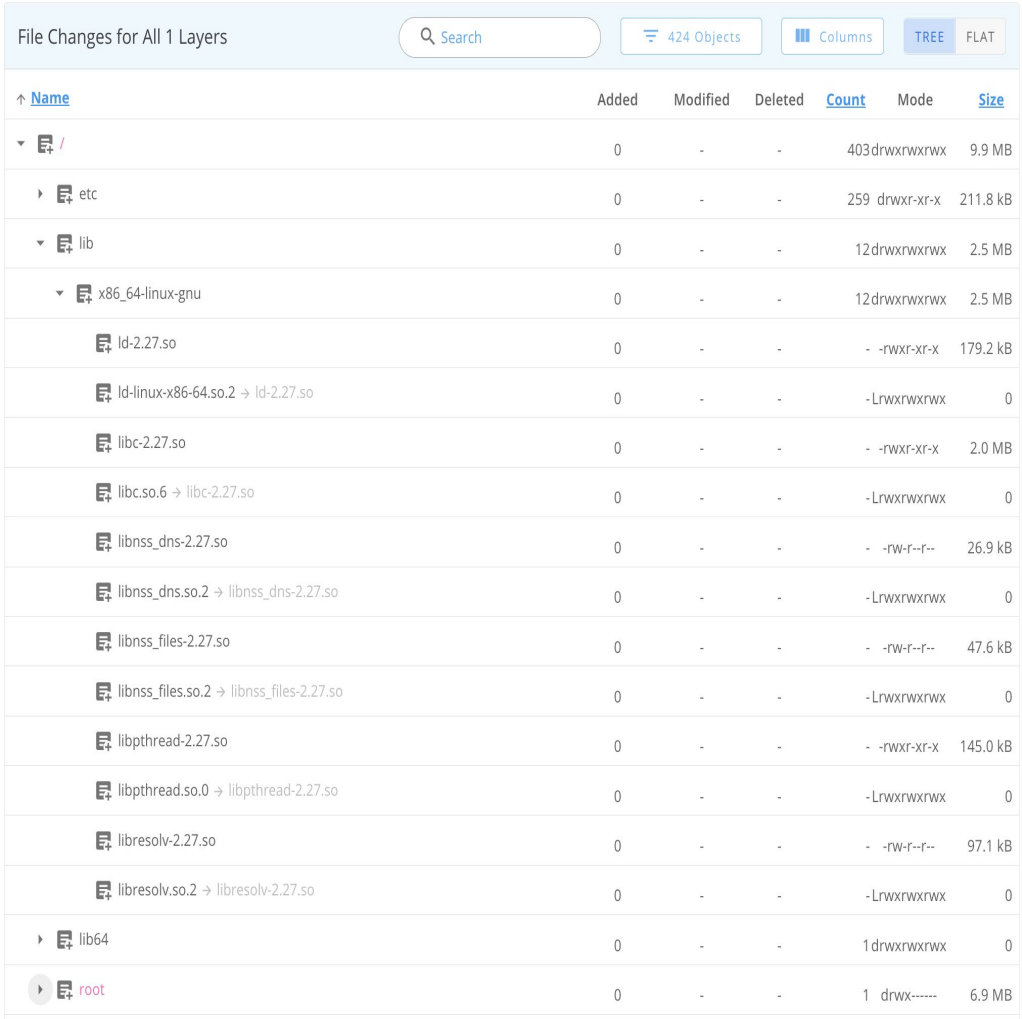
```
FROM node:18 as builder

WORKDIR /usr/src/app
COPY app/package*.json ./
RUN npm install
COPY app/ .

FROM gcr.io/distroless/nodejs:18

WORKDIR /usr/src/app
COPY --from=builder /usr/src/app .
EXPOSE 8080
CMD [ "node", "server.js" ]
```

distroless app - 166.8MB

- distroless/nodejs++
- COPY “everything”

Create Minimal Container Images with Slim (the easy way)





Name	Added	Modified	Deleted	Count	Mode	Size
/	0	-	-	403	drwxrwxrwx	9.9 MB
etc	0	-	-	259	drwxr-xr-x	211.8 kB
lib	0	-	-	12	drwxrwxrwx	2.5 MB
x86_64-linux-gnu	0	-	-	12	drwxrwxrwx	2.5 MB
ld-2.27.so	0	-	-	-	-rwxr-xr-x	179.2 kB
ld-linux-x86-64.so.2 → ld-2.27.so	0	-	-	-	Lrwxrwxrwx	0
libc-2.27.so	0	-	-	-	-rwxr-xr-x	2.0 MB
libc.so.6 → libc-2.27.so	0	-	-	-	Lrwxrwxrwx	0
libnss_dns-2.27.so	0	-	-	-	-rw-r--r--	26.9 kB
libnss_dns.so.2 → libnss_dns-2.27.so	0	-	-	-	Lrwxrwxrwx	0
libnss_files-2.27.so	0	-	-	-	-rw-r--r--	47.6 kB
libnss_files.so.2 → libnss_files-2.27.so	0	-	-	-	Lrwxrwxrwx	0
libpthread-2.27.so	0	-	-	-	-rwxr-xr-x	145.0 kB
libpthread.so.0 → libpthread-2.27.so	0	-	-	-	Lrwxrwxrwx	0
libresolv-2.27.so	0	-	-	-	-rw-r--r--	97.1 kB
libresolv.so.2 → libresolv-2.27.so	0	-	-	-	Lrwxrwxrwx	0
lib64	0	-	-	1	drwxrwxrwx	0
root	0	-	-	1	drwx------	6.9 MB

slim/go/app - 9.9MB

- App with deps
- Binary and non-binary deps included automatically
- **/root/go/src/api/api** - 6.9 MB
- https://github.com/docker-slim/examples/tree/master/3rdparty/mux-go-api

```
FROM node:18

WORKDIR /usr/src/app
COPY app/package*.json ./
RUN npm install
COPY app/ .

EXPOSE 8080
CMD [ "node", "server.js" ]
```

slim app - 90.9MB

- fat(1GB) -> slim(90MB)
- auto-“scratch”

Kubernetes Debugging Techniques that Don't Work with Slim Images (and Why)

- **Embedded** debugging tools
- kubectl cp
- kubectl **exec**

Requires tar to be there in the container, else it will fail

`kubectl cp`

Copy files and directories to and from containers

`Kubectl exec` → Needs a shell to be there in the container bash/sh

`Debugging utilities` → htop,curl etc

Minimal Container Images: “kubectl exec” failure demo

```
% export PNAME=`kubectl get pods -l app=kubeday-demo -o jsonpath='{.items[0].metadata.name}'`
% kubectl exec -it -c app $(PNAME) -- bash

error: Internal error occurred: error executing command in container: failed to exec in container:
failed to start exec "f53291c5312b9186c9a67adb1c350da51030c1f44e8f8d2e5d3c9430128f3a4f": OCI runtime
exec failed: exec failed: unable to start container process: exec: "bash": executable file not found in
$PATH: unknown
```

Ephemeral Containers: Background

- Why/what?
- History
- How they are implemented in Kubernetes
- How they are exposed through **kubectl debug**

Ephemeral containers

These are the containers to help debug pods where there is no way debug directly using exec etc.

It attaches the container to the same pod and share a few namespaces as well

No readiness, liveness probes
Resources is disallowed

created using ephemeralcontainers API

You cannot expose PORT

Ephemeral containers

kubectl debug -it nginx --image busybox -- /bin/sh

↓

Adds a new container with busybox image to nginx image

↓

Use same resources as the pods

↓

helps to debug containers without any debugging utilities

```
ephemeralContainers:
- command:
  - /bin/sh
  image: busybox
  imagePullPolicy: Always
  name: debugger-cnwvf
  resources: {}
  stdin: true
  terminationMessagePath: /dev/termination-log
  terminationMessagePolicy: File
  tty: true
```

- cgroup
- mnt
- pid

```
    ephemeralContainerStatuses:
    - containerID: containerd://5659598ca268827dc60472336958afd5e5ebc88ac563c8aadf41660bf5eae26e
      image: docker.io/library/busybox:latest
      imageID: docker.io/library/busybox@sha256:fcd85228d7a25feb59f101ac3a955d27c80df4ad824d65f5757a954831450185
```

Debugging Capabilities

- **Pod level** debugging
  - **same** pod
  - pod **copy** for advanced debugging
    - pod copy as a replacement for the target pod
    - customizing debugged container properties
- **Node level** debugging

Simple kubectl debug Demo

```
% export PNAME=`kubectl get pods -l app=kubeday-demo -o jsonpath='{.items[0].metadata.name}'`
% kubectl debug -it -c debug-sidecar-bbox --image busybox --target app ${PNAME}

Targeting container "app". If you don't see processes from this container it may be because the
container runtime doesn't support this feature.
If you don't see a command prompt, try pressing enter.

/ # ps aux
PID   USER     TIME  COMMAND
    1 root      0:00 node server.js
   13 root      0:00 sh
   18 root      0:00 ps aux

/ # echo $$
13

/ # ls
bin   dev   etc   home  proc  root  sys   tmp   usr   var

/ # ls /proc/1/root
bin    dev    etc    lib    lib64  proc   run    sys    tmp    usr    var
```

Ephemeral Containers Without "kubectl debug"

- internals/APIs
- additional capabilities (e.g., setting security context, mounting volumes)
- curl examples

Gotchas

- can’t remove ephemeral containers
- not all container properties are available
- process namespace sharing gotchas
- **kubectl debug** gotchas (can’t set security context or mount volumes)

Ephemeral Containers: Support

- kubernetes version 1.23(beta), 1.25(GA)
- Ephemeral containers support in different hosted Kubernetes services:
  - Civo Kubernetes
  - GKE (Google)
  - EKS (AWS)
  - AKS (Microsoft/Azure)
  - DOKS (DigitalOcean)

Debugging Container Images

- Custom Debugging Images:
  - **Netshoot**
  - **KoolKits** by LightRun (jvm, node, python, go)
- Do It Yourself:
  - Dockerfile <- stuff :-)
  - http://**nixery**.dev (do it yourself, but on demand)

Koolkit/Netshoot/Nixery kubectl debug Demos

```
export PNAME=`kubectl get pods -l app=kubeday-demo -o jsonpath='{.items[0].metadata.name}'`
kubectl debug -it -c debug-sidecar-kk --image lightruncom/koolkits:node --target app ${PNAME}

Targeting container "app". If you don't see processes from this container it may be because the
container runtime doesn't support this feature.
If you don't see a command prompt, try pressing enter.

root@kubeday-demo-657564c8c9-g5wdw:/usr/local/bin# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.8 597528 51148 ?        Ssl  06:09   0:00 node server.js
root        38  0.1  0.1   7944  7264 pts/0    Ss   06:51   0:00 /bin/bash
root       205  0.0  0.0   5904  2844 pts/0    R+   06:52   0:00 ps aux
```

```
export PNAME=`kubectl get pods -l app=kubeday-demo -o jsonpath='{.items[0].metadata.name}'`
kubectl debug -it -c debug-sidecar-netshoot --image nicolaka/netshoot --target app ${PNAME}

Targeting container "app". If you don't see processes from this container it may be because the
container runtime doesn't support this feature.
If you don't see a command prompt, try pressing enter.

                    dP            dP                           dP
                    88            88                           88
88d888b. .d8888b. d8888P .d8888b. 88d888b. .d8888b. .d8888b. d8888P
88'  `88 88ooood8   88   Y8ooooo. 88'  `88 88'  `88 88'  `88   88
88    88 88.  ...   88         88 88    88 88.  .88 88.  .88   88
dP    dP `88888P'   dP   `88888P' dP    dP `88888P' `88888P'   dP

Welcome to Netshoot! (github.com/nicolaka/netshoot)

 kubeday-demo-657564c8c9-g5wdw ▶ ~ ▶
```

```
export PNAME=`kubectl get pods -l app=kubeday-demo -o jsonpath='{.items[0].metadata.name}'`
kubectl debug -it -c debug-sidecar-nix --image nixery.dev/shell/which/lsof/ps/iproute2/netcat-
gnu/tshark/tcpdump/curl/jq/nodejs --target app ${PNAME}

Targeting container "app". If you don't see processes from this container it may be because the
container runtime doesn't support this feature.
If you don't see a command prompt, try pressing enter.

bash-5.1# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
0            1  0.0  0.8 597528 51148 ?        Ssl  06:09   0:00 node server.js
0          265  0.1  0.0   4852  4008 pts/0    Ss   07:02   0:00 bash
0          271  0.0  0.0   7104  2480 pts/0    R+   07:02   0:00 ps aux

bash-5.1# ls
bin  dev  etc  include  lib  libexec  nix  nix-support  proc  sbin  share  sys  tmp  usr
```

Key Takeaways

- **Minimal container images on k8s are ready for mainstream use**!!!

- Ephemeral containers make it possible to debug Scratch, Slim and Distroless app images in production.

- Use tools like Nixery, Netshoot, Koolkits for faster debugging

Thank You!

- **Demo** repo:
  - *https://github.com/kubeday-japan/demo*

- Try **ephemeral** containers on Civo!

- Create **minimal container images** with docker-slim or Slim!

- And thanks to **Akihiro Suda** for creating **nerdctl**!!!