KubeCon CloudNativeCon Europe 2021

Virtual

Sidecars at Netflix

From Basic Injection to First Class Citizens

Rodrigo Campos - Kinvolk
Manas Alekar - Netflix

About us

Rodrigo

Software Engineer, **Kinvolk**

Github: rata
Email: rodrigo@kinvolk.io,
rodrigo@sdfg.com.ar

Manas

Software Engineer, **Netflix**

Github: Netflix

Agenda

Netflix

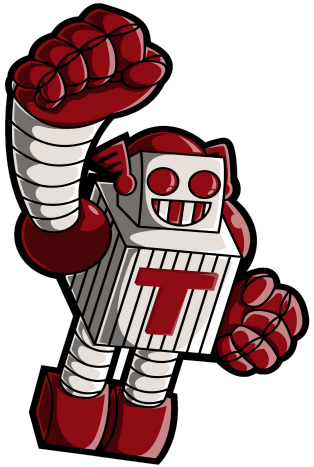
- Migration to containers
- System Services
- Titus Executor
- Adopting Kubernetes
- Challenges with Kubelet

Kubernetes

- Problems we have
- Efforts made to improve the situation

Compute Migration

Titus is predominant



Spinnaker

Special uses for VMs

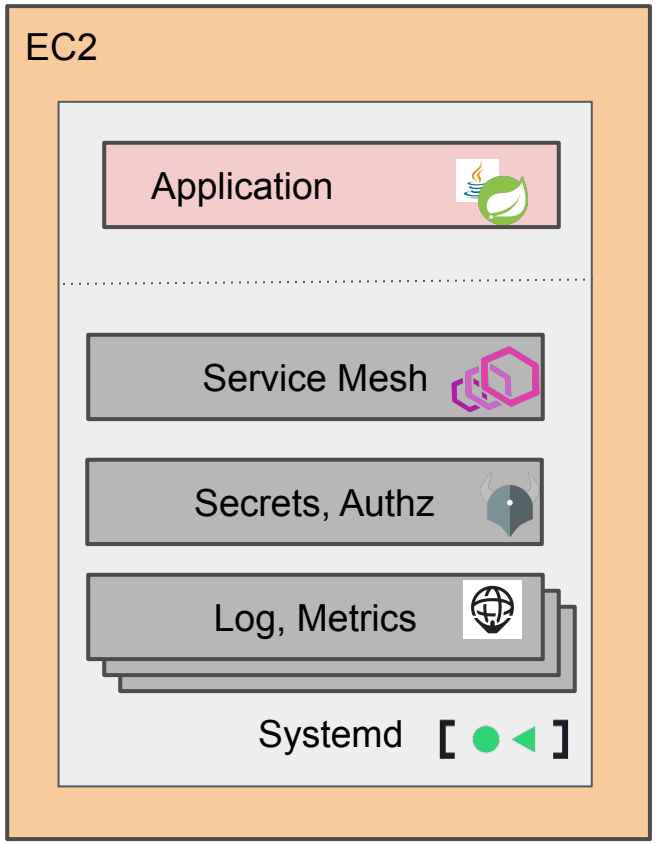
Why ?

- Flexible Infrastructure
  - Scale
  - Efficiency

When ?

- Within 2 years
  - Automatic migration

VMs



What runs in a VM?

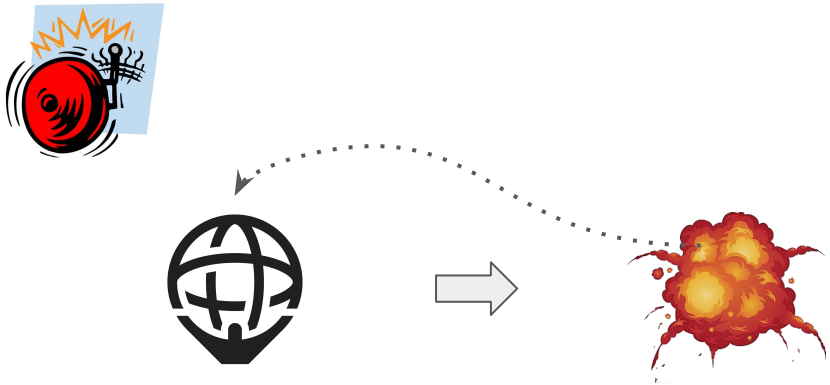
- The application, typically on the JVM
- Supporting daemons on host
  - RPC: Service Mesh
  - Security: Secrets, Authz
  - Observability: Logs, Metrics
- Basic Features
  - SSH, Postfix, etc.

Daemons

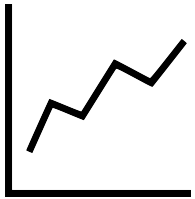
Metrics

Certificates

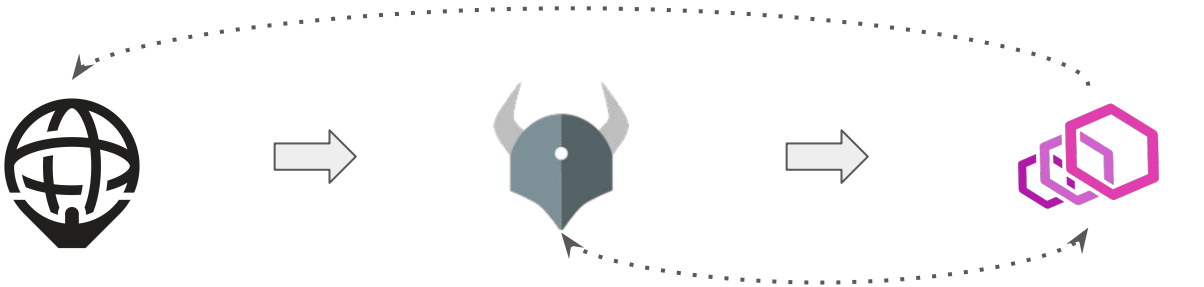
Daemons



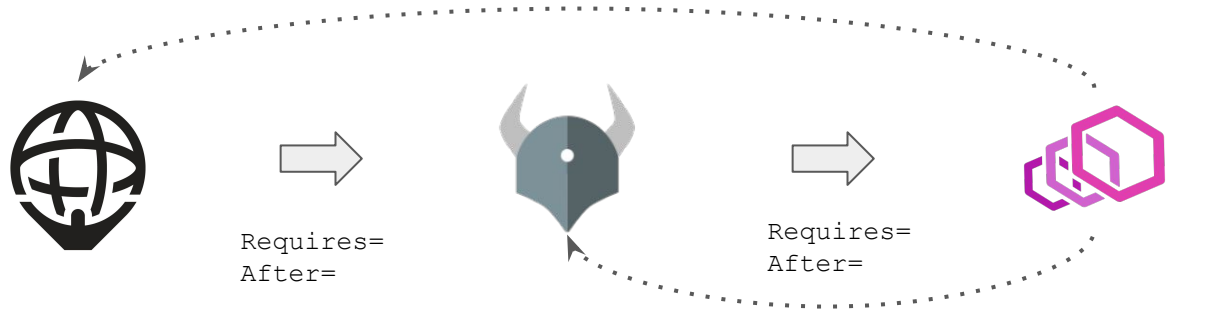
Metrics



Daemons



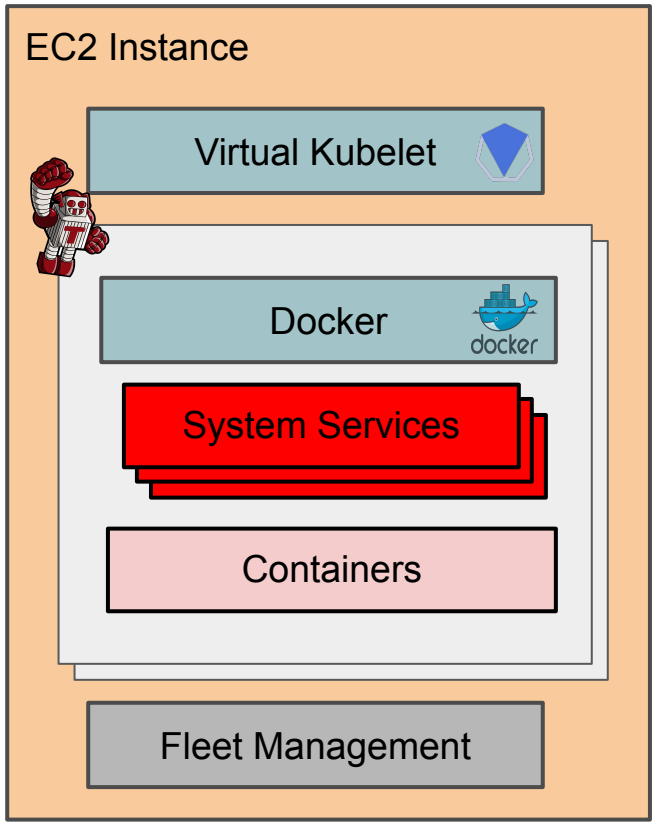
Daemons




```
Restart=always
```

Logs

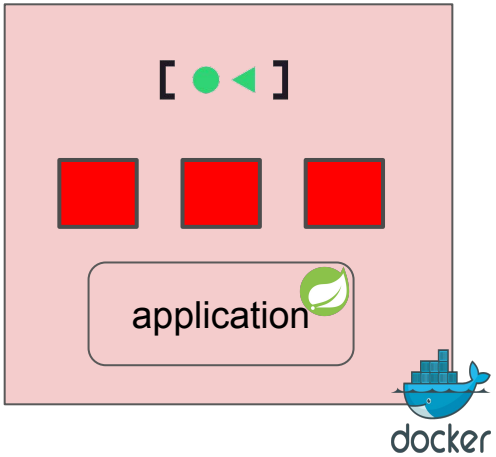
Titus Executor



What runs on Titus Host?

- User containers run on a multi tenant AWS host instance
- Supporting service are provided by system services on Titus
- Agents needed to manage the fleet

Why System Services?



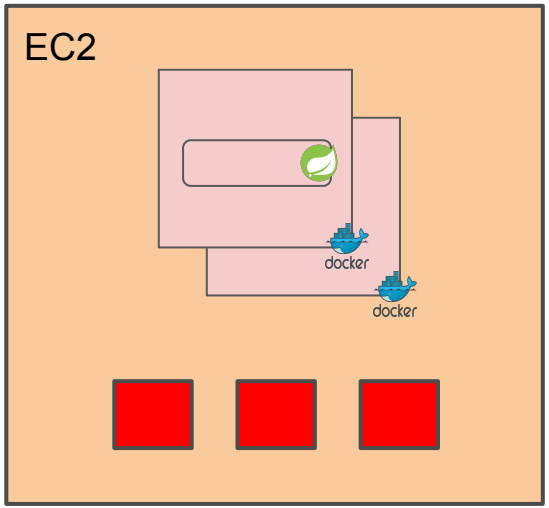
\+ Simplest implementation to get feature parity

\- Affects how fast we can upgrade the fleet
ex: CVE, upgrades

\- Cannot just run an image from docker hub

\- Cannot secure advanced system services
ex: Metadata Service, Storage

Why System Services?



\+ No major ordering issues once services are running

\- Resources and fault domains are shared by containers

\- We have workloads with SLAs we this model cannot meet
ex: stable configuration once running, no disruptions

\- It is difficult to secure multi tenant services !

We provide `http://agent:80/$container/`

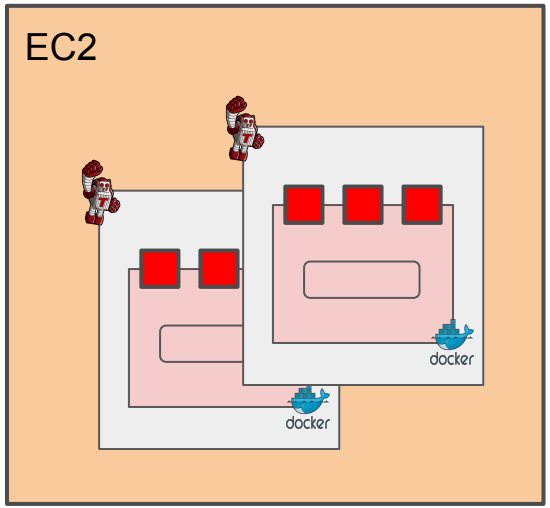
to view logs for a container....

`../../../etc/passwd`

We provide `/volumes/metrics/common_tags`

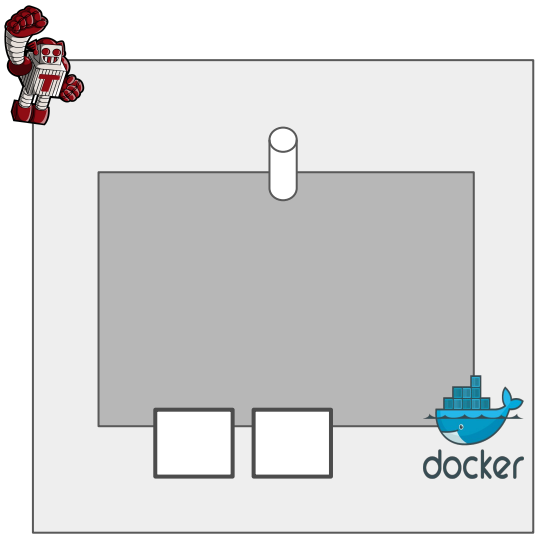
to configure metrics agent ...

Why System Services?



- Every container gets its own system services
- We tie the system service to the container's
  - Lifetime
  - Identity
  - Security
  - Resources

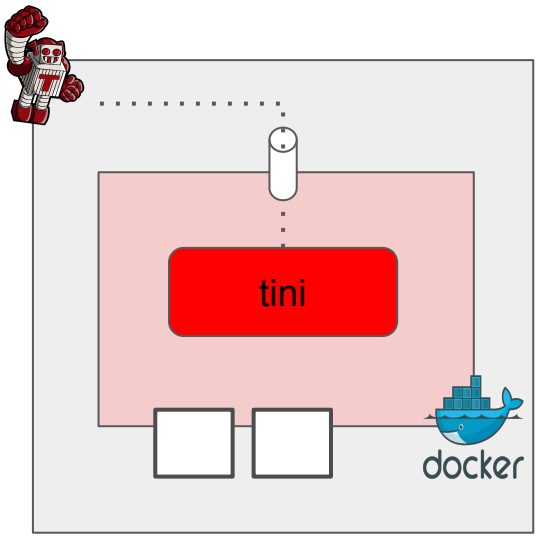
Container Lifecycle



Create Container

1. Virtual kubelet launches titus executor
2. Creates the container using docker
3. Bind mounts system services
4. And a unix socket to control startup

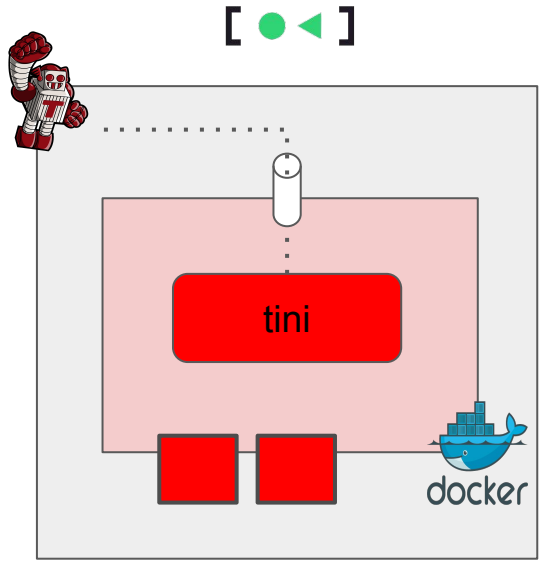
Container Lifecycle



Run Container

- Start the container, tini blocks for executor
  tini = tiny init

Container Lifecycle



Start System Services

- Start the container, tini blocks for executor
  tini = tiny init
- Start the system service
  - In required order
  - With restart policies

Systemd


```
[Unit]
Description=Titus logviewer for container %i
ConditionPathIsDirectory=/var/lib/titus-inits/%i/ns

StartLimitIntervalSec=30
StartLimitBurst=5
CollectMode=inactive-or-failed
PartOf=titus-container@%i.target

[Service]
EnvironmentFile=/var/lib/titus-environments/%i.env
# Run as root (UID 0, GID 0) and with CAP_DAC_OVERRIDE so that containers with a `USER` instruction work
ExecStart=/usr/bin/runc --root /var/run/docker/runtime-${TITUS_OCI_RUNTIME}/moby exec --user 0:0 --cap CAP_DAC_OVERRIDE --cwd
/titus/adminlogs ${TITUS_CONTAINER_ID} bin/adminlogs

Restart=on-failure
RestartSec=3
KillMode=mixed
```

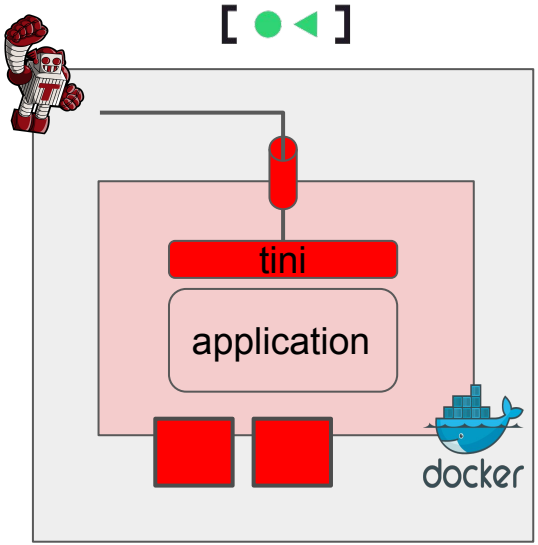
Systemd


```
[Unit]
Description=Titus logviewer for container %i
ConditionPathIsDirectory=/var/lib/titus-inits/%i/ns


StartLimitIntervalSec=30
StartLimitBurst=5
CollectMode=inactive-or-failed
PartOf=titus-container@%i.target


[Service]
EnvironmentFile=/var/lib/titus-environments/%i.env
# Run as root (UID 0, GID 0) and with CAP_DAC_OVERRIDE so that containers with a `USER` instruction work
ExecStart=/usr/bin/runc --root /var/run/docker/runtime-${TITUS_OCI_RUNTIME}/moby exec --user 0:0 --cap CAP_DAC_OVERRIDE --cwd
/titus/adminlogs ${TITUS_CONTAINER_ID} bin/adminlogs


Restart=on-failure
RestartSec=3
KillMode=mixed
```

Systemd


```
[Unit]
Description=Titus logviewer for container %i
ConditionPathIsDirectory=/var/lib/titus-inits/%i/ns


StartLimitIntervalSec=30
StartLimitBurst=5
CollectMode=inactive-or-failed
PartOf=titus-container@%i.target


[Service]
EnvironmentFile=/var/lib/titus-environments/%i.env
# Run as root (UID 0, GID 0) and with CAP_DAC_OVERRIDE so that containers with a `USER` instruction work
ExecStart=/usr/bin/runc --root /var/run/docker/runtime-${TITUS_OCI_RUNTIME}/moby exec --user 0:0 --cap CAP_DAC_OVERRIDE --cwd
/titus/adminlogs ${TITUS_CONTAINER_ID} bin/adminlogs


Restart=on-failure
RestartSec=3
KillMode=mixed
```

Container Lifecycle



Run Entrypoint

- Release tini using the socket
- Tini remains PID1 in most containers
  .. and redirects stdout/ stderr
- Allows more advanced capabilities

Advanced Injection

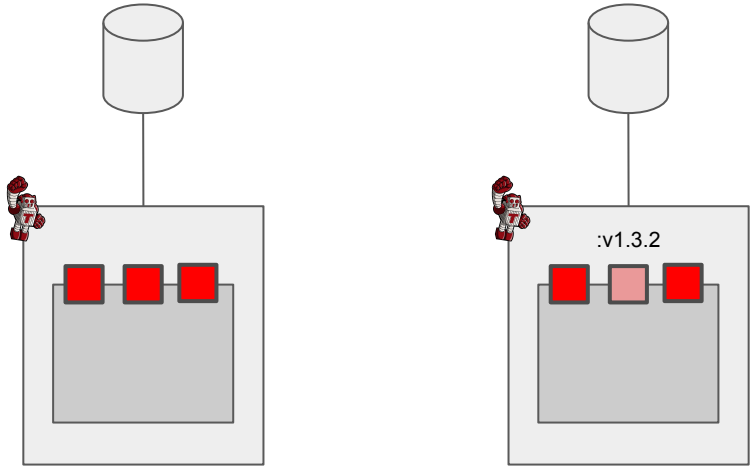

```
[Unit]
Description=SSHD for container %s
ConditionPathIsDirectory=/var/lib/titus-inits/%i/ns
...

[Service]
Environment=TITUS_PID_1_DIR=/var/lib/titus-inits/%i
# TSA is used in the main container processes to handle certain syscalls
# To ensure a sane user experience, we want sshd-spawned processes to
# get TSA help too. This environment variable ensures that will happen.
Environment=TITUS_NSENTER_USE_TSA=true
EnvironmentFile=/var/lib/titus-environments/%i.env
ExecStart=/apps/titus-executor/bin/titus-nsenter /titus/sshd/usr/sbin/sshd -D -e
LimitNOFILE=65535
...
```

SSH


```
char *pid1dir = getenv(TITUS_PID_1_DIR);
int titus_pid_1_fd = open(pid1dir, O_RDONLY | O_CLOEXEC);

int apparmor_fd;
char apparmor_profile[8192];
get_apparmor_profile(titus_pid_1_fd, apparmor_profile, &apparmor_fd));

populate_namespaces(titus_pid_1_fd, namespace_fds);

for (i = 0; i < ARRAY_SIZE(namespaces); i++) {
        setns(namespace_fds[i], namespaces[i].nstype);
}

set_up_apparmor(apparmor_profile, apparmor_fd);

if (vfork() == 0) {
        prctl(PR_SET_PDEATHSIG, SIGKILL);
        execvp(argv[1], &argv[1]);
```

System Services

Versioning



Latest

Pinned

System Services

In Summary

\+ Easy to implement using *rootless* data containers

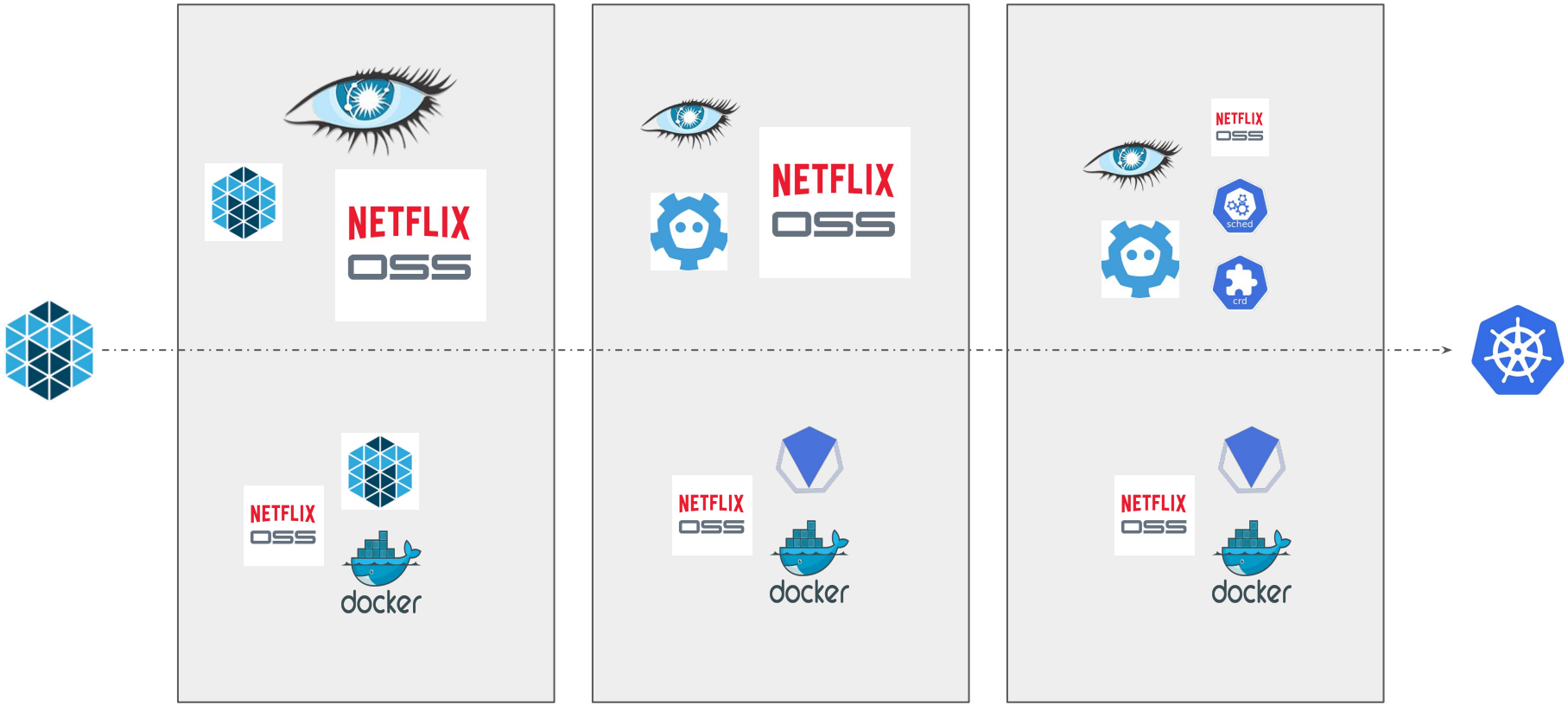
\+ Shared `mountns` for sshd, shared lifetimes via `pidns`

\- Can only have one user container and many services

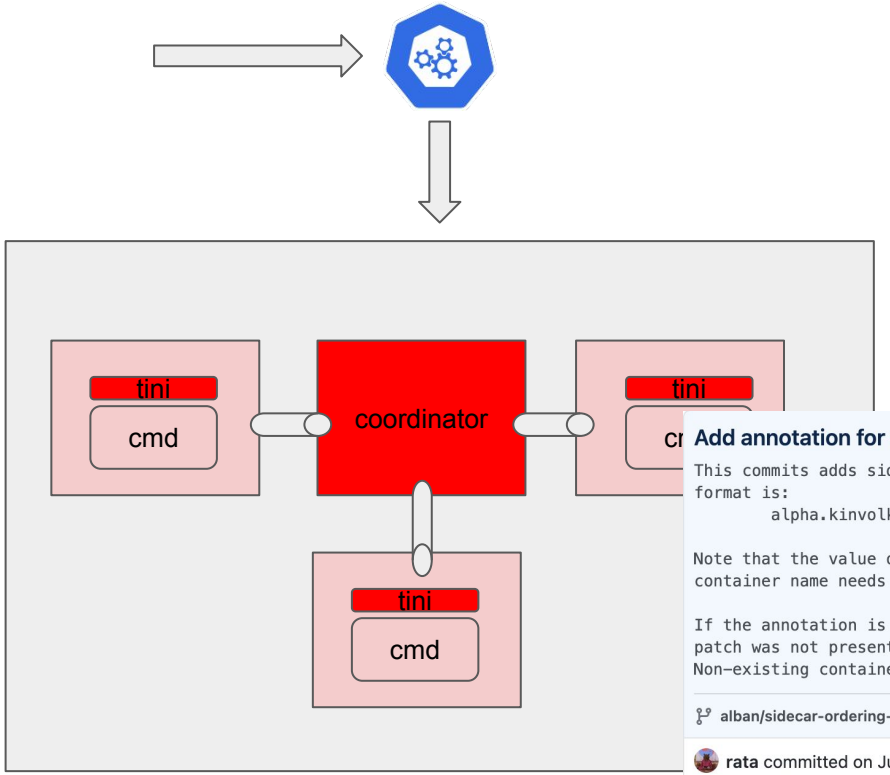
\- Terrible usability unless the sidecar authors use C or golang

\- Difficult to package debug symbols/ tools with the container

Kubernetes Adoption



Options ?



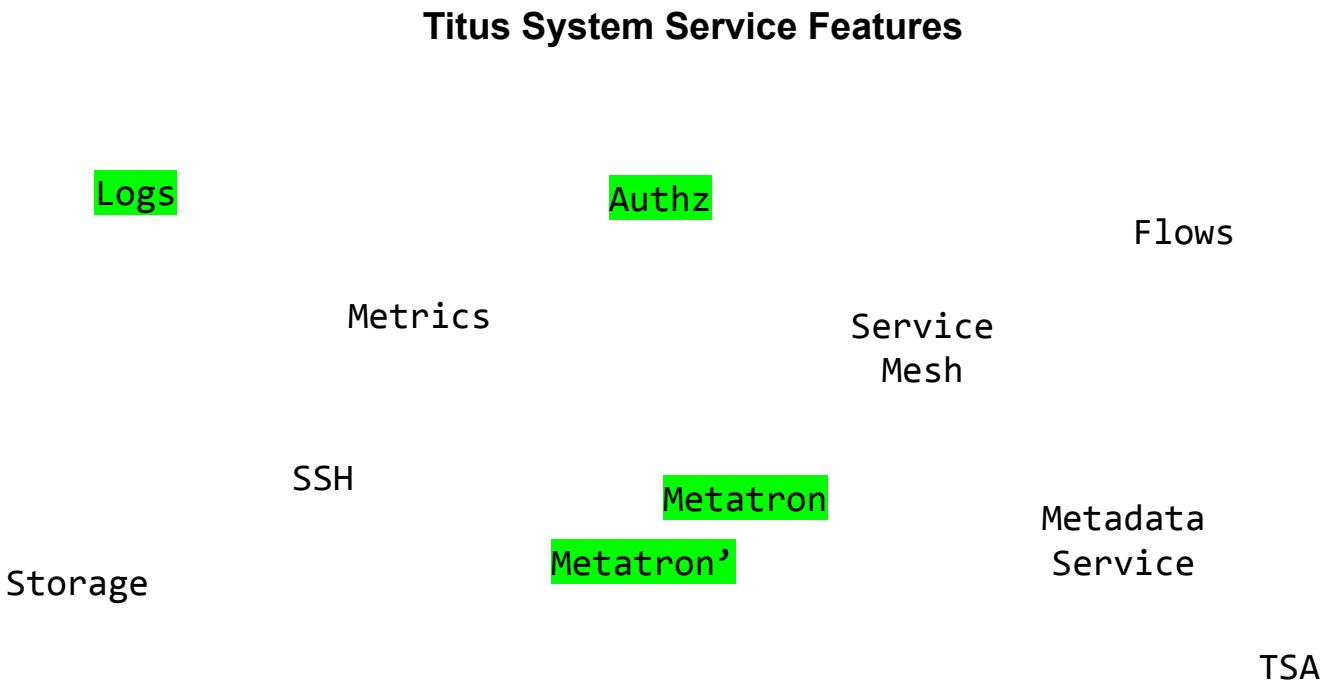
Add annotation for sidecar ordering

```
This commits adds sidecar containers support using a pod annotation. The
format is:
       alpha.kinvolk.io/sidecar: '["container-1", "container-2"]'

Note that the value of the annotation needs to be valid JSON, so each
container name needs to be enclosed with double quotes.

If the annotation is not present or malformed, the behavior is as this
patch was not present: no contianer is considered a sidecar.
Non-existing containers referenced in the annotation are ignored.
```

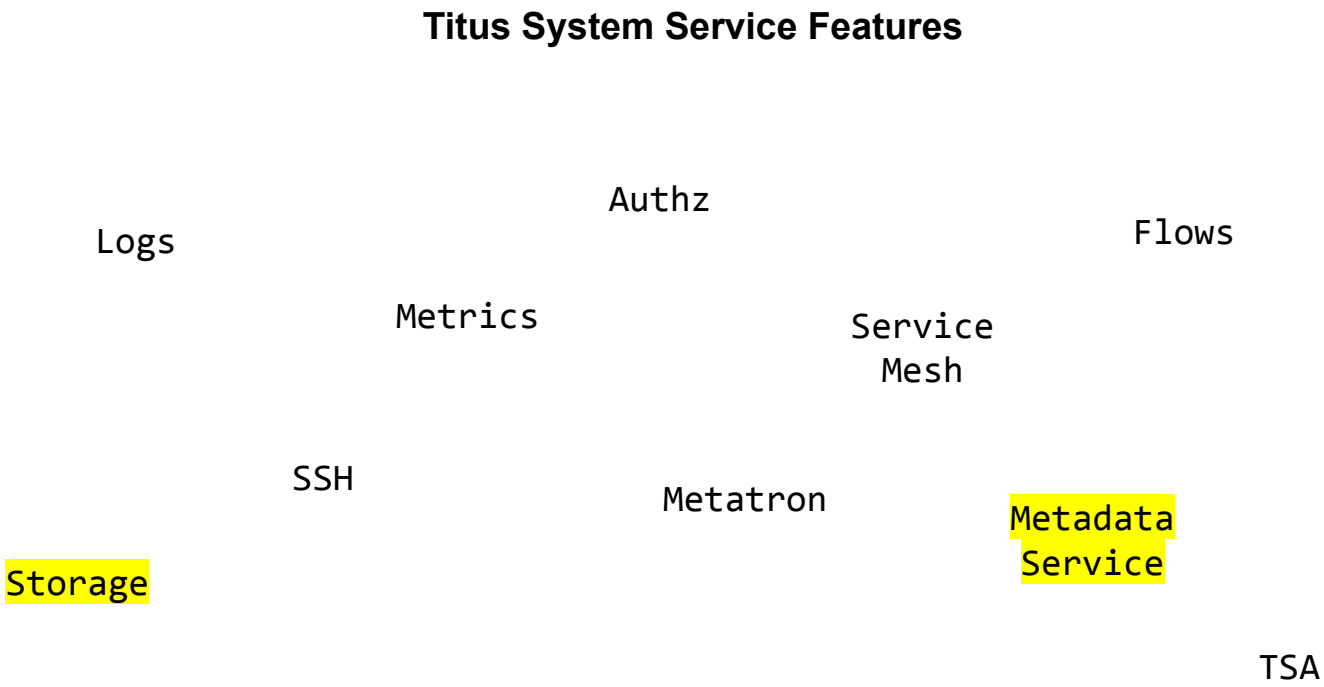
alban/sidecar-ordering-tests-2 + mauricio/userns-suggestions + rata/sidecar-ordering-tests

rata committed on Jun 11, 2020

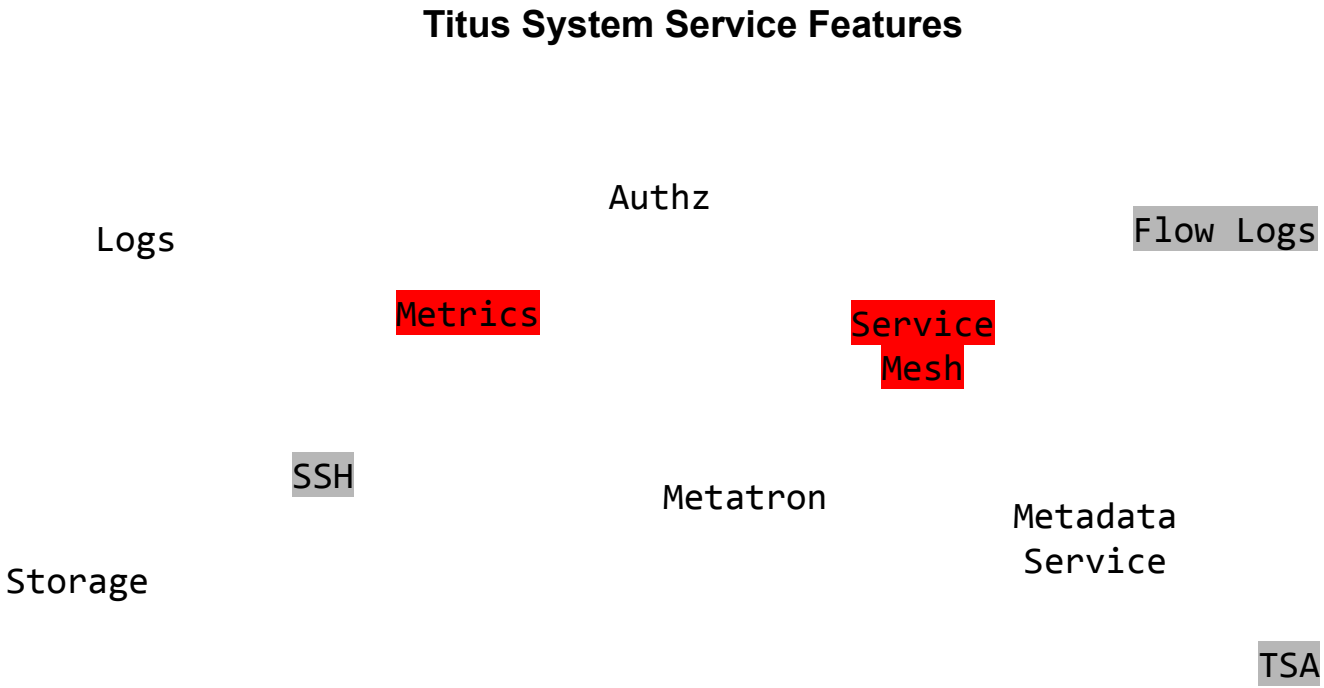
System Services



System Services



System Services



Titus Executor & Kubelet

Some Challenges

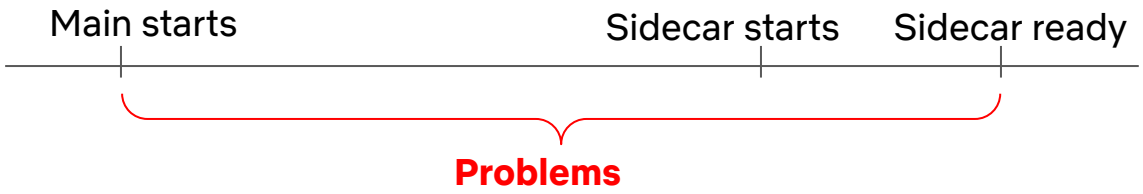
- We run services in a operationally sensitive fashion
  - Letting them crash is not a good solution
  - We expect stable configuration when deployed
- We run services with an aggressive security posture
  - userns, seccomp, apparmor
- We write single tenant system services
  - Resource isolation is important, this cannot regress
  - Restricting failure domain has stood the test of time
- We want to move towards more composable workloads

Sidecars in K8s upstream

- Kubernetes is like Jon Snow
  - Knows nothing
  - All containers are treated in the same way
- Let’s see the problems that we have today...

What are the problems?

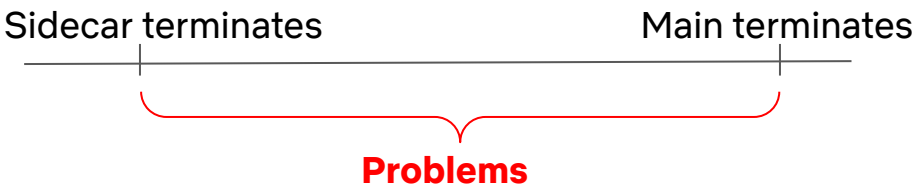
- No startup order guarantee



- Workarounds
  - Wait for containers with a wrapper, like linkerd-await
  - Let them crash until they eventually converge
    - Becomes super slow when you add more and more sidecars

What are the problems?

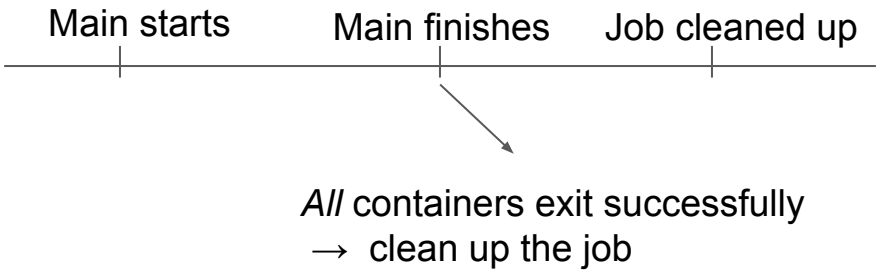
- No shutdown order guarantee

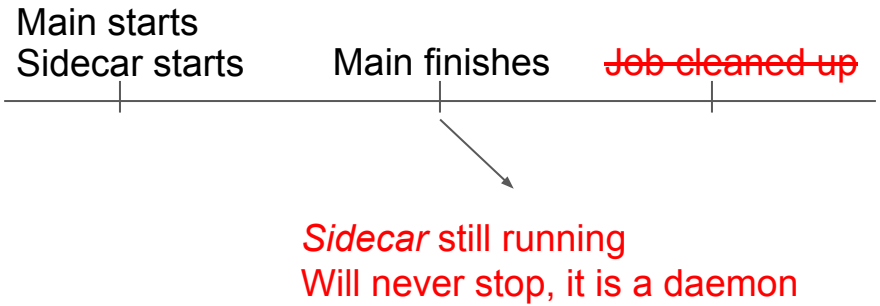


- Some workarounds
  - preStop: sleep 30 or similar
    - SIGTERM is sent after preStop hooks execute
    - Make preStop hook take a long time, so it is not killed
    - Time to kill a pod is greatly increased on 100% of the cases
  - Ignore SIGTERM or just lose traffic in some cases

What are the problems?

- Can’t use them with jobs (without hacks)





- Can’t use them with initContainers either
  - Not possible to workaround!

KEP 753: sidecar containers

- A little bit of history
  - Created by Joseph Irving in **2018**!
  - Started a **big** discussion
  - I joined forces with Joseph in May 2020 and iterated the KEP
    - Several companies are using a K8s fork for sidecar containers
  - October 2020: KEP is rejected. Start from scratch
- Decided to explore radically different ideas
  - The KEP proposed small changes, so it didn't make sense anymore
  - Some ideas in the horizon:
    - Pod phases/runlevels - ala sysVinit
    - "DependsOn" semantics - ala systemd

Sidecars from scratch

- Start from scratch - collect use cases
  - Thanks everyone adding use cases!
- Waiting for more pre-proposals/pre-KEP
  - Tim Hockin created a pre-proposal. Pod lifecycle phases
  - Choose one proposal to turn into a KEP
- We made good progress
- We are at this pre-proposal stage
  - Joseph moved on
  - I don’t have time to champion this new effort from pre-proposal to KEP and GA
  - Matthias Bertschy will continue to move this forward

Summary

- Sidecar usage in the wild
  - There are several problems today
  - Some companies are forking Kubernetes for this functionality only
  - Big win for others too
- Sidecars in Kubernetes upstream
  - At pre-proposal stage
  - A KEP is expected out of pre-proposals

Reference material

- Kubernetes upstream:
  - Sidecar KEP - https://bit.ly/2PzM9UC
  - Sidecar use cases doc - https://bit.ly/31r1vxk
  - Pre-proposal by Tim Hockin - http://bit.ly/thockin-pod-lifecycle-phases-proposal
  - Linkerd-await - https://github.com/olix0r/linkerd-await
  - Russian Doll: Extending Containers with Nested Processes - Christie Wilson & Jason Hall, Google - https://www.youtube.com/watch?v=iz9_omZ0ctk
- Sidecars at Netflix
  - Titus Executor https://github.com/Netflix/titus-executor

Thanks!

Questions?

KubeCon CloudNativeCon Europe 2021

Virtual