Microsoft Azure

Tilting a Pyramid

Confidentiality in a Cloud Native Environment

Magnus Kulke

2023-02-05; FOSDEM 23

Hi!

- SWE in Microsoft‘s kinvolk team
- kinvolk‘s focus is Open Source, Linux and Containers (👉 Flatcar Container Linux, Inspector Gadget, Headlamp)
- Exploring ways to integrate Confidential Computing technology with Containers

github.com/mkulke
linkedin.com/in/magnuskulke

Caveat! ☝

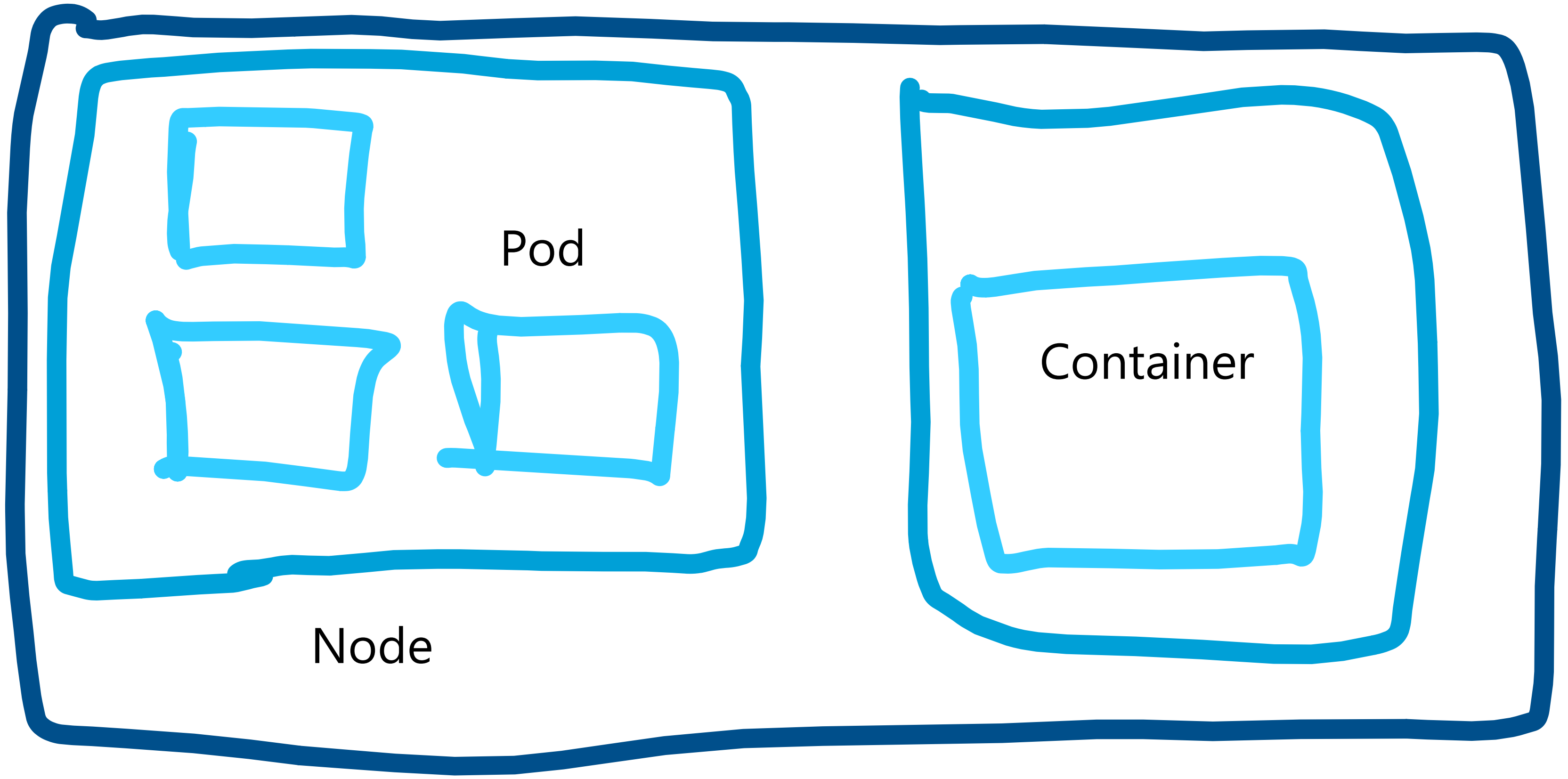
- There will be generalization and simplifications, sorry about that!
- Not a comprehensive coverage of the state of the art of Confidential Containers
- Things are very much evolving
- Idea is: providing pointers for folks who want to get involved

Cloud Native?

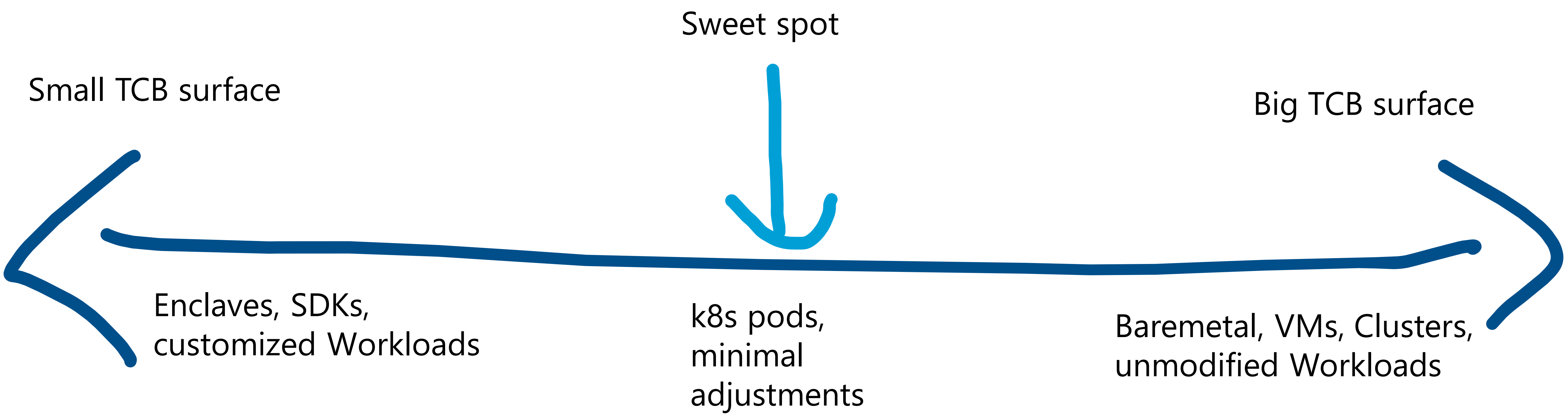
- Ecosystem of practices, tools and interfaces that ease deployment and management of applications on cloud platforms
- Cloud native apps can leverage *IaaS*, *PaaS*, *CaaS*, or *FaaS*
- Containers are the most prominent CN technology today
- *Kubernetes* has been adopted as the go-to solution for container orchestration and mgmt

Kubernetes (k8s)

- Operating Kubernetes is not trivial, hosted offerings by CSPs are popular
- *Pods:* a logical environment (isolated, resource-constrained) composed of individual containers



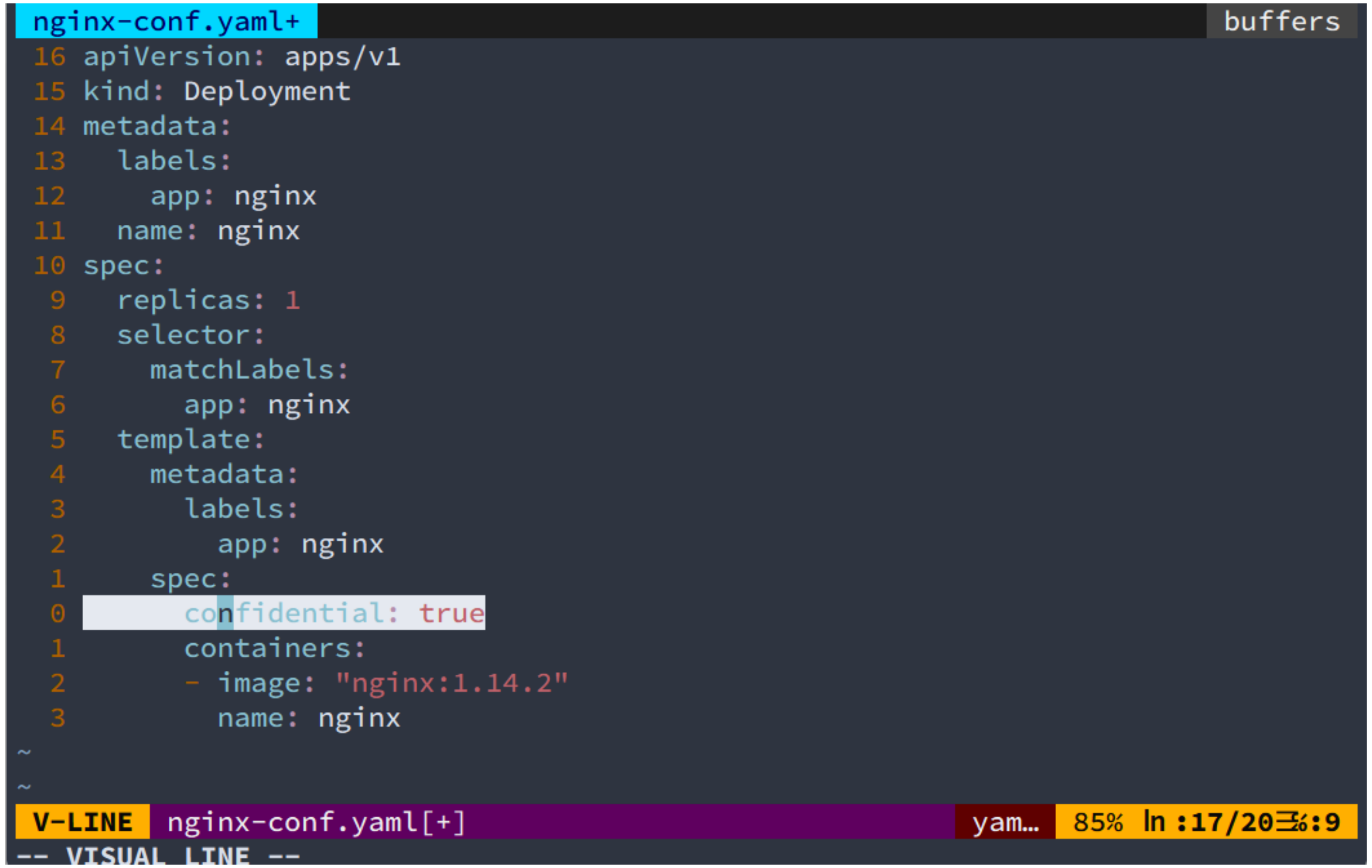
TCB trade-off


Sweet spot
Small TCB surface
Big TCB surface
Enclaves, SDKs, customized Workloads
k8s pods, minimal adjustments
Baremetal, VMs, Clusters, unmodified Workloads

CC ♥ K8S?

- Some workloads are locked out of CN (and public clouds), due to compliance
- Ease adoption of Conf. Computing by enabling confidentiality with minimal upfront investments
- k8s has been widely adopted by the industry
- k8s provides abstractions and technology which we can leverage for CC
- Example: SEV & TDX leverage VMs to provide a TEE => Kata Containers is a proven solution for isolating Pods in a VM

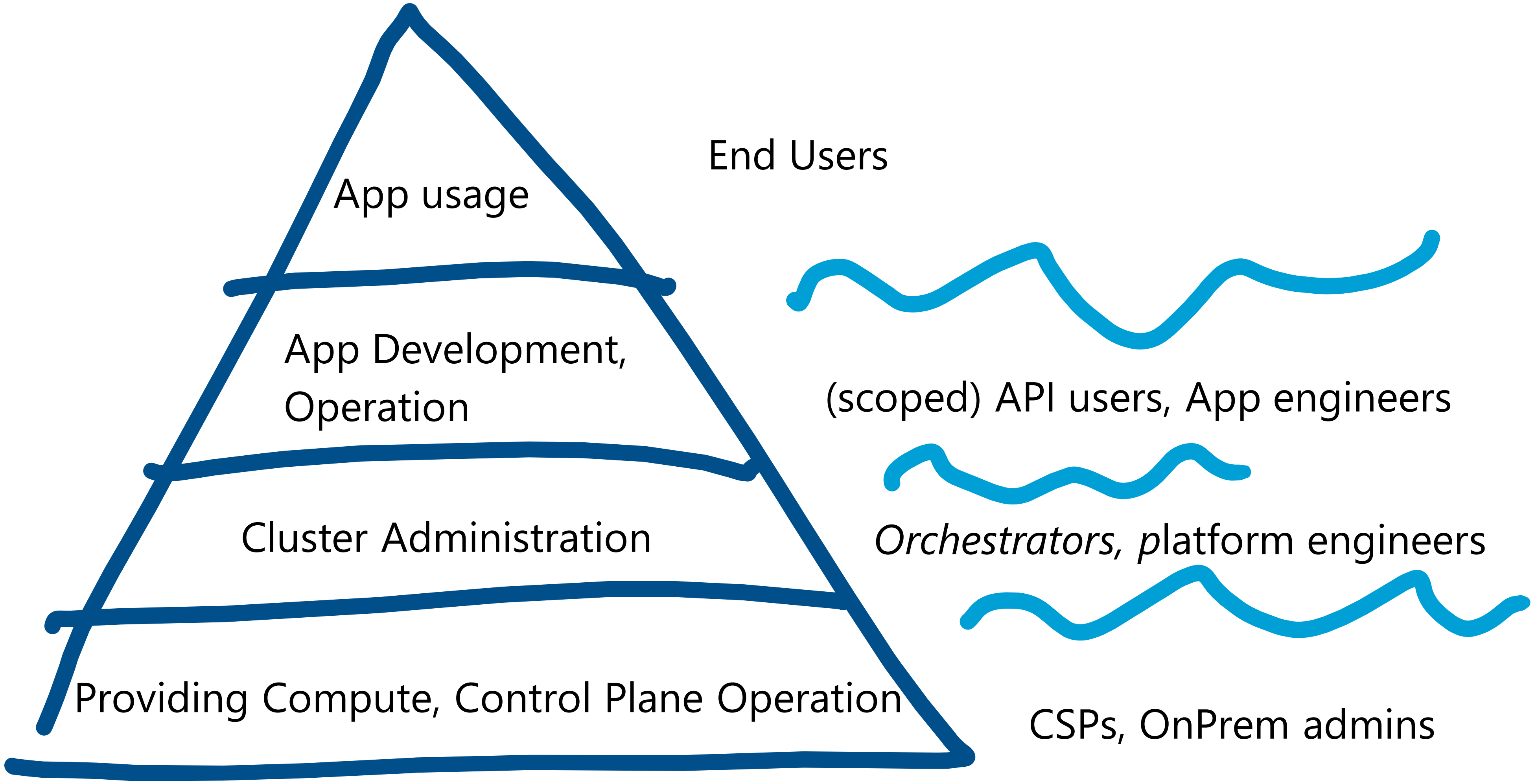
Ideally:



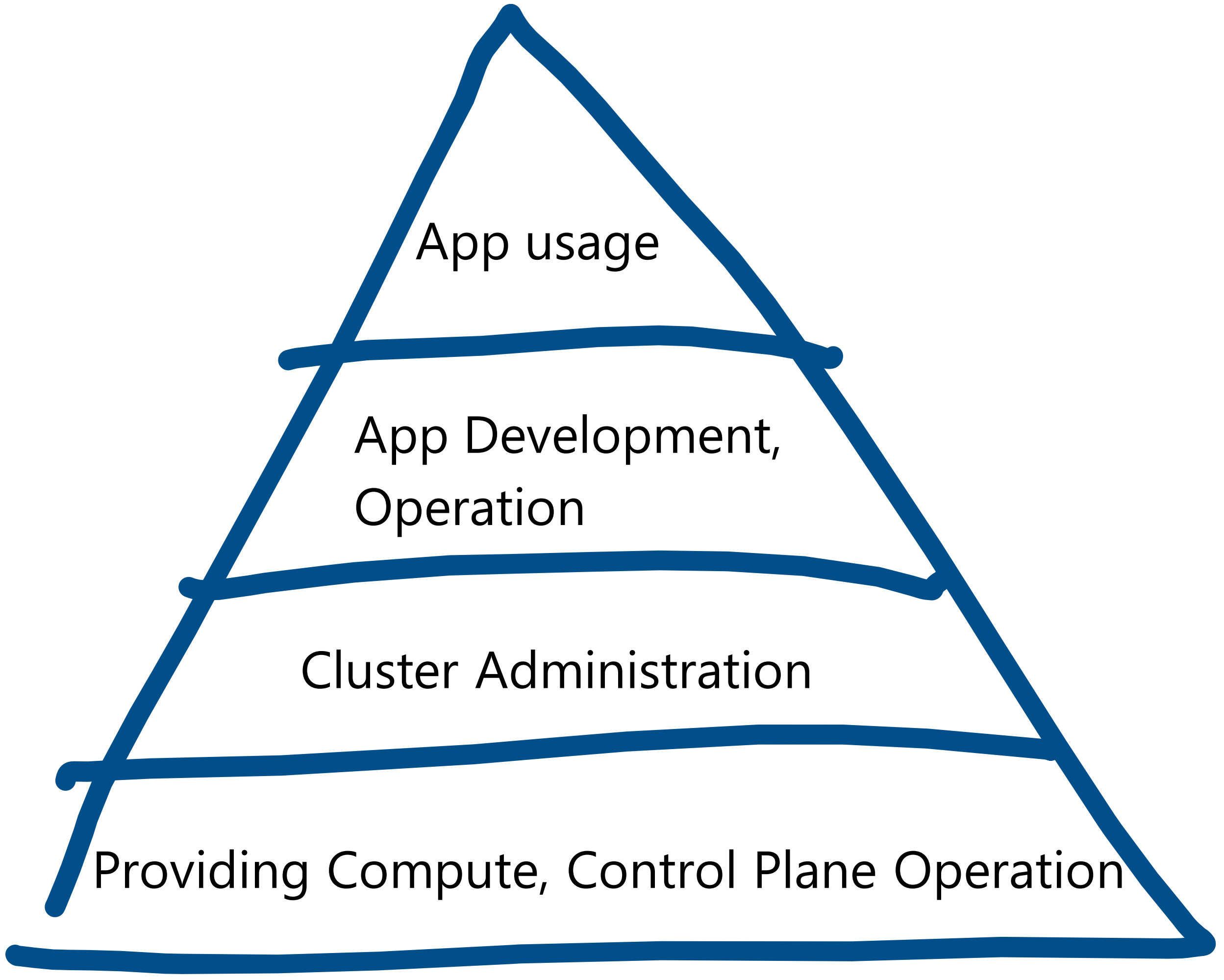
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: nginx
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      confidential: true
      containers:
      - image: "nginx:1.14.2"
        name: nginx
~
~
```

```
V-LINE  nginx-conf.yaml[+]                                  yam…  85%  ln :17/20☰%:9
-- VISUAL LINE --
```

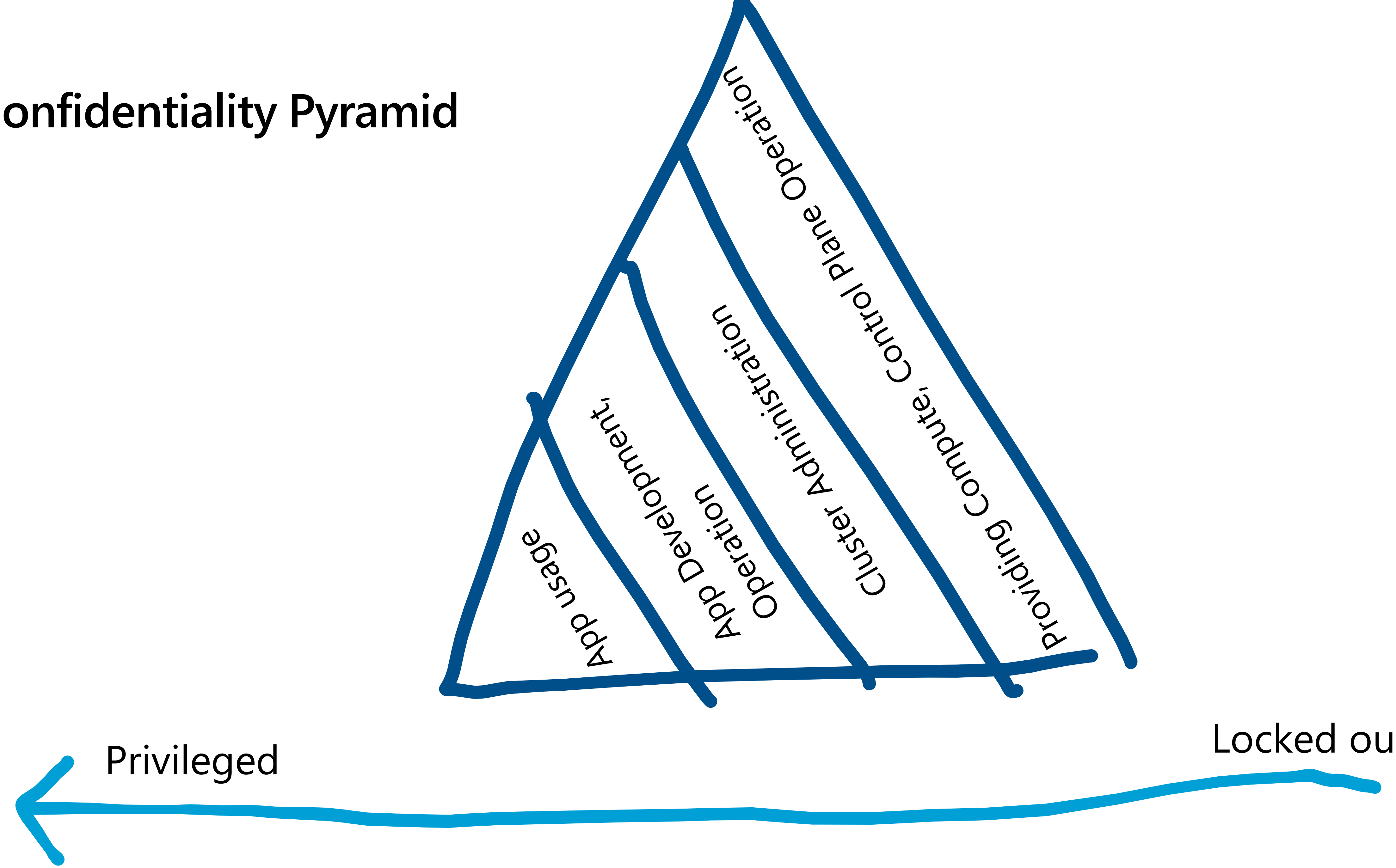
K8s Privilege Pyramid


App usage
End Users
App Development, Operation
(scoped) API users, App engineers
Cluster Administration
Orchestrators, platform engineers
Providing Compute, Control Plane Operation
CSPs, OnPrem admins

Confidentiality Pyramid


App usage
App Development, Operation
Cluster Administration
Providing Compute, Control Plane Operation

Confidentiality Pyramid


Providing Compute, Control Plane Operation
Cluster Administration
App Development, Operation
App usage
Privileged
Locked out

Multiple Challenges

- OCI image management
- Avoiding metadata interference
- Trusted Control Plane
- …

Images

- Image lifecycle is managed by the infrastructure layers (e.g. kubelet, containerd)
- In a TEE we need verified or encrypted images for our workloads
- There are OCI facilities for both, but we need to move logic into the TEE

Images, cont.

- Pragmatic bandaid: Pull images in encrypted memory (tmpfs) on a Confidential Pod VM.
- Downside: we need to provision potentially big chunks of memory
- Downside: Pods cannot share images or image layers

- We can create an encrypted scratch space for image-pull to require less memory, but unshared images will still yield bad start-up time

- Alternative: We can stream (encrypted) image layers (or otherwise chunked up blocks) from the host to a Confidential Pod
- The technology (containerd remote snapshotters) is not 100% meeting the requirements yet, but it's pretty close

Metadata

- k8s will transform a specified workload in multiple ways (e.g. mount points, envs, image definitions)
- This is inherent to k8s: e.g. for service discovery, provisioning secrets or enforcing compliance (think: „ubuntu:latest“ image is replaced on-the-fly w/ a vetted digest)

- We don‘t want the control plane tampering with our trusted workloads, we want to run exactly what we specified
- Example: A Confidential Pod includes a local Redis cache in the TEE, a control-plane transparently injecting a REDIS_HOST env would topple confidentiality

Metadata, cont.

- Possible Solution: container technology within the TEE can review the delta between user-specified and to-be-provisioned spec
- We can validate whether we‘re fine with the applied changes in user-specified policies
- Downside: there are inherent „dynamisms“ in k8s pods, policies to catch those are potentially complex
- Downside: not-ideal ux, deviation from „confidential: true“ simplicity target

- Variation: Log changes that have been performed and have a trusted component acknowledge the changes as valid before running a workload

Control Plane

- Users, tools interact with k8s Pods via Control Plane APIs and host components, which are not part of the TEE
- Observability is a requirement for many cloud native workloads
- We need to obscure Logs, Traces, Metrics from the non-trusted parties
- We need to de-privilege the non-trusted parties to prevent them from executing commands in the scope of a Confidential Pod

Control Plane, cont.

- Pragmatic bandaid: Lock down problematic parts of the API in the TEE.
- Downside: Not practical in the long run

- Potential solution: Split container management APIs into infrastructure & trusted parts
- Operate a „Trusted Control Plane“ that users/tool interact with for Confidential Pods (ideally transparent to user)
- Downside: Large effort

- Alternative solution: Have encrypted transport between privileged user and the container management tech in the TEE
- Downside: Very invasive change, need to extend many parts of the container stack

Summary

- Confidential Computing and Cloud Native Containers are a good match
- There's hairy questions to resolve for reconciling both worlds
- It's exciting, chime in: https://github.com/confidential-containers

Thx!