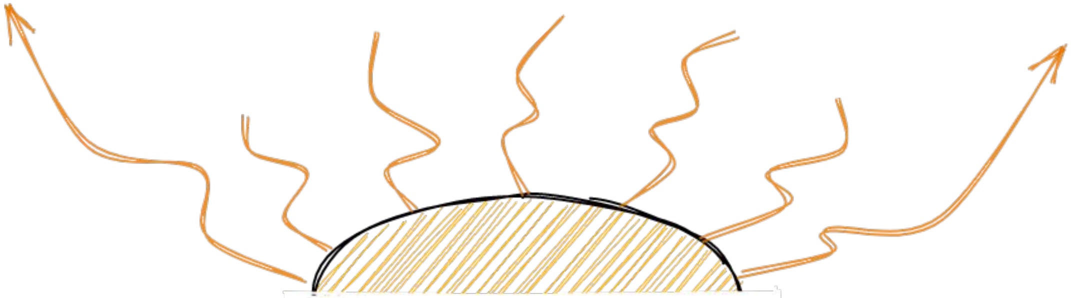
RED HAT®
OPENSHIFT



The Dawn of OpenShift sandboxed containers

Adel Zaalouk
azaalouk@redhat.com

RED HAT®
OPENSHIFT

Introduction + Use-Cases

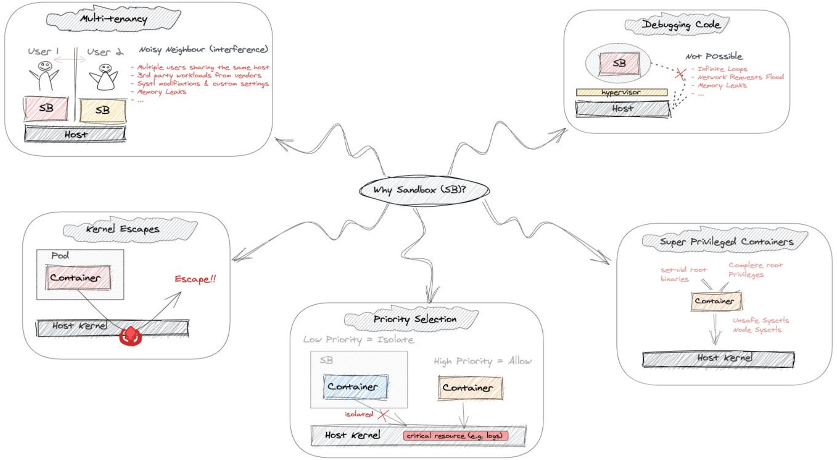
Sandboxding

Vegas Mode

Trade-offs

When / Where

Sandboxing??



A sandbox is a tightly controlled environment where programs run [1]

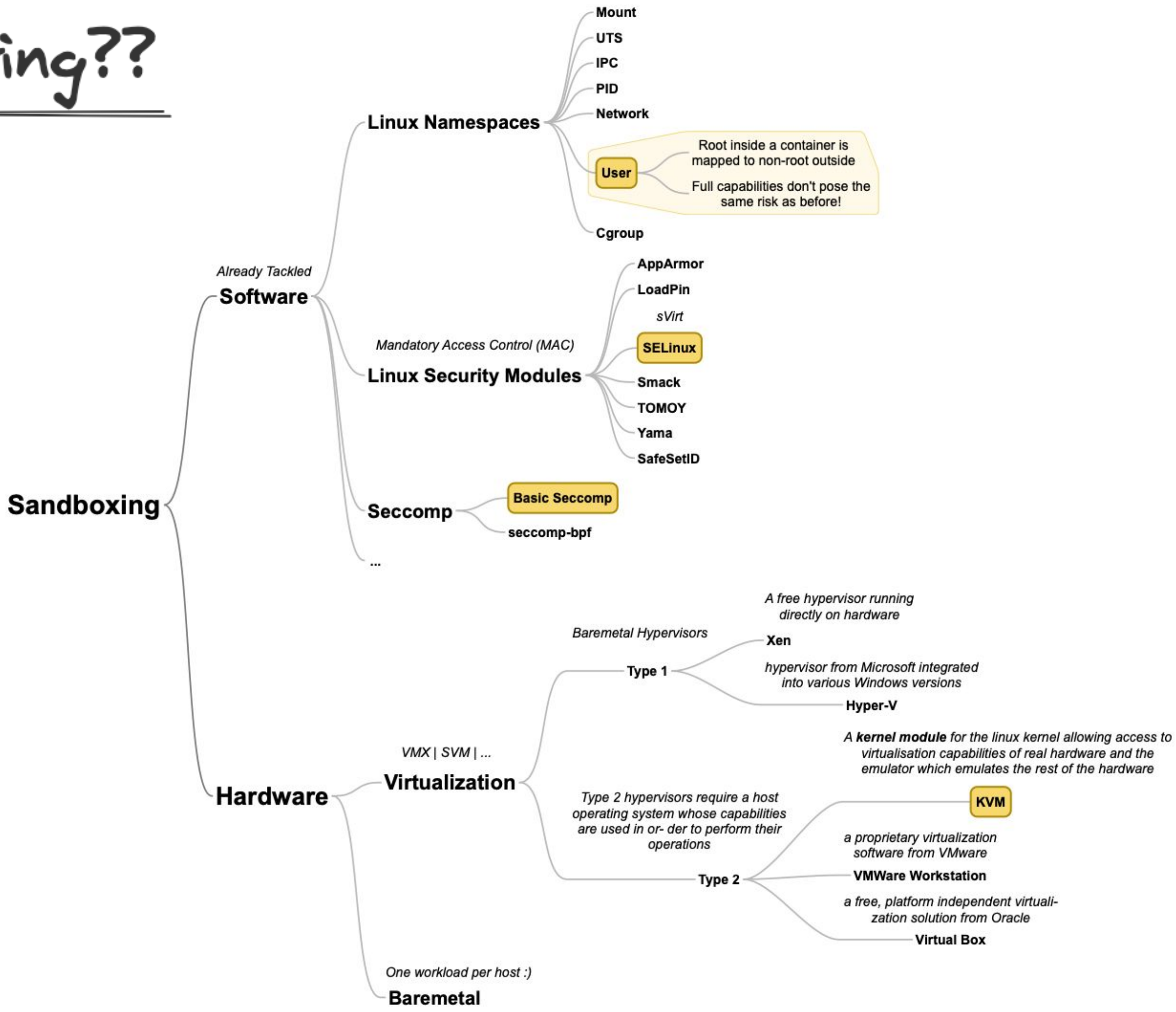
Environments that impose irrevocable restrictions on resource usage [2]

It is often used to execute untested or untrusted programs or code without risking harm to the host machine or operating system [3]

Keeps your program isolated from the rest of the system, by using any one of the different methods available in the Linux kernel [4]

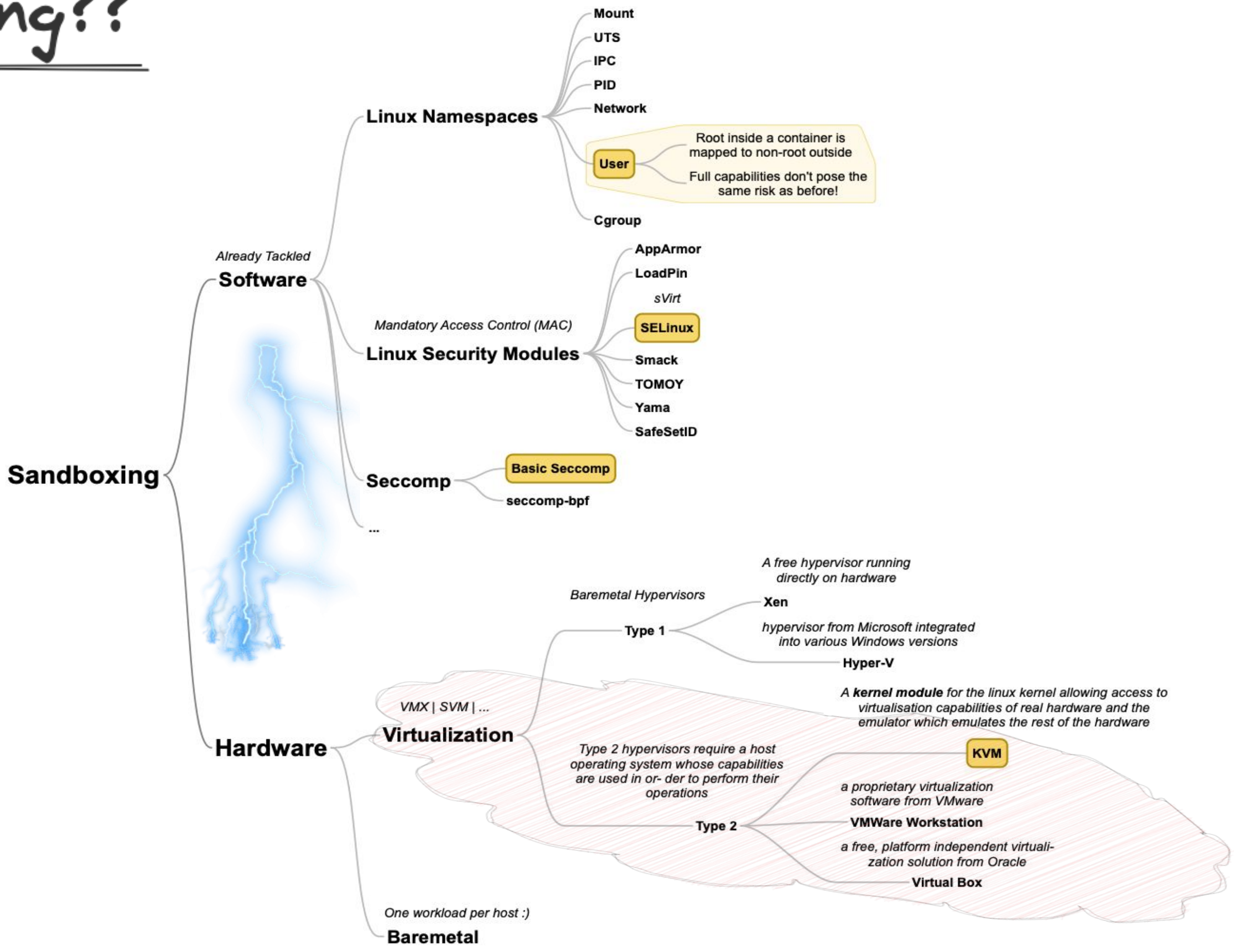
...

Sandboxing??



- Sandboxing
  - *Already Tackled* Software
    - Linux Namespaces
      - Mount
      - UTS
      - IPC
      - PID
      - Network
      - User
        - Root inside a container is mapped to non-root outside
        - Full capabilities don't pose the same risk as before!
      - Cgroup
    - *Mandatory Access Control (MAC)* Linux Security Modules
      - AppArmor
      - LoadPin
      - *sVirt* SELinux
      - Smack
      - TOMOY
      - Yama
      - SafeSetID
    - Seccomp
      - Basic Seccomp
      - seccomp-bpf
    - ...
  - Hardware
    - *VMX | SVM | ...* Virtualization
      - *Baremetal Hypervisors* Type 1
        - *A free hypervisor running directly on hardware* Xen
        - *hypervisor from Microsoft integrated into various Windows versions* Hyper-V
      - *Type 2 hypervisors require a host operating system whose capabilities are used in or- der to perform their operations* Type 2
        - *A **kernel module** for the linux kernel allowing access to virtualisation capabilities of real hardware and the emulator which emulates the rest of the hardware* KVM
        - *a proprietary virtualization software from VMware* VMWare Workstation
        - *a free, platform independent virtualization solution from Oracle* Virtual Box
    - *One workload per host :)* Baremetal

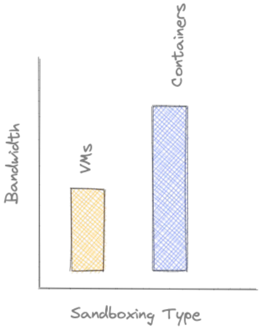
Sandboxing??

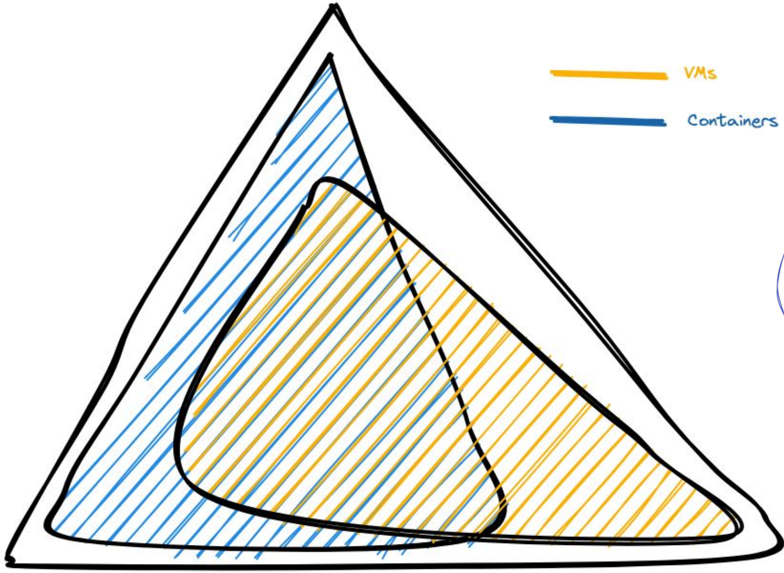


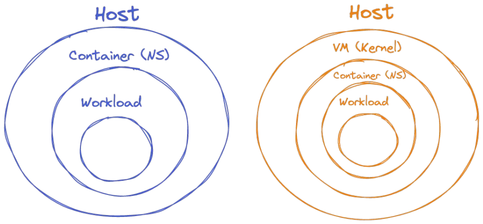
- Sandboxing
  - *Already Tackled*
  - Software
    - Linux Namespaces
      - Mount
      - UTS
      - IPC
      - PID
      - Network
      - User
        - Root inside a container is mapped to non-root outside
        - Full capabilities don't pose the same risk as before!
      - Cgroup
    - *Mandatory Access Control (MAC)*
    - Linux Security Modules
      - AppArmor
      - LoadPin
      - *sVirt*
      - SELinux
      - Smack
      - TOMOY
      - Yama
      - SafeSetID
    - Seccomp
      - Basic Seccomp
      - seccomp-bpf
    - ...
  - Hardware
    - *VMX | SVM | ...*
    - Virtualization
      - *Baremetal Hypervisors*
      - Type 1
        - *A free hypervisor running directly on hardware*
        - Xen
        - *hypervisor from Microsoft integrated into various Windows versions*
        - Hyper-V
      - *Type 2 hypervisors require a host operating system whose capabilities are used in or- der to perform their operations*
      - Type 2
        - *A **kernel module** for the linux kernel allowing access to virtualisation capabilities of real hardware and the emulator which emulates the rest of the hardware*
        - KVM
        - *a proprietary virtualization software from VMware*
        - VMWare Workstation
        - *a free, platform independent virtuali- zation solution from Oracle*
        - Virtual Box
    - *One workload per host :)*
    - Baremetal

Trade-offs

Efficiency





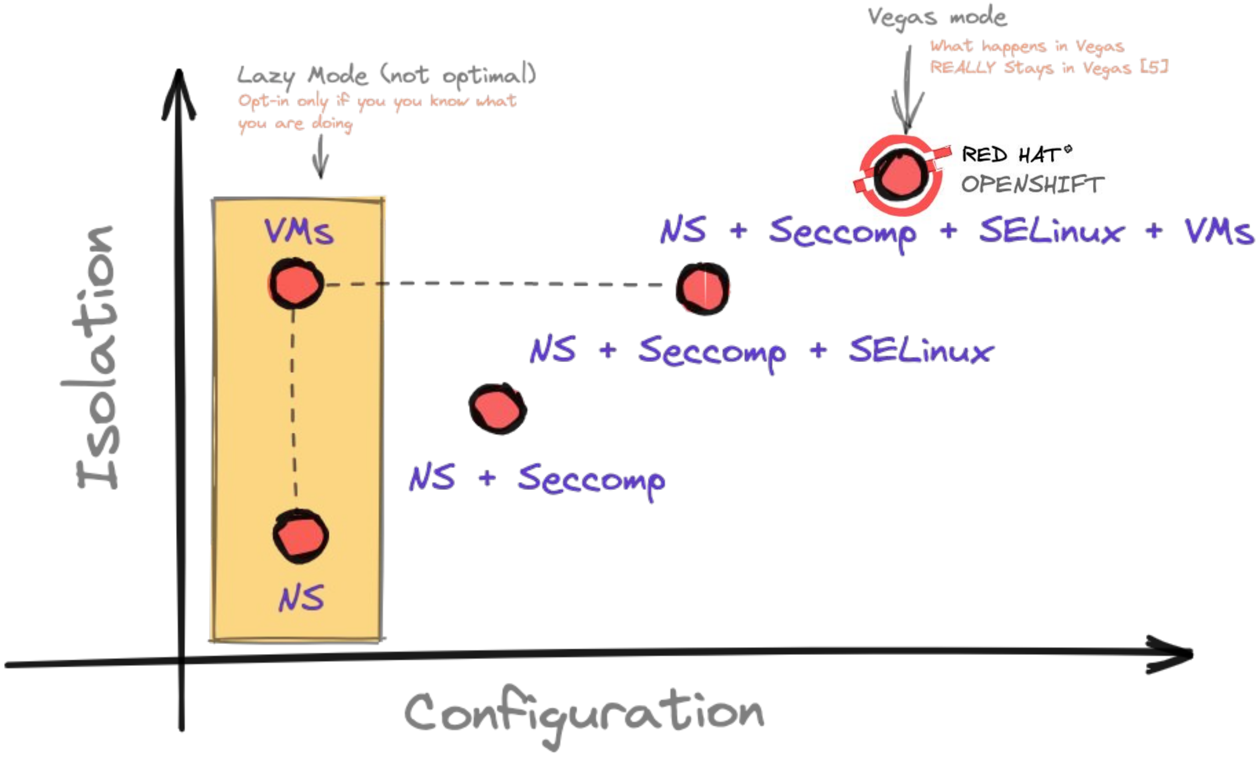


e.g., Kernel

Isolation

Performance

Vegas Mode



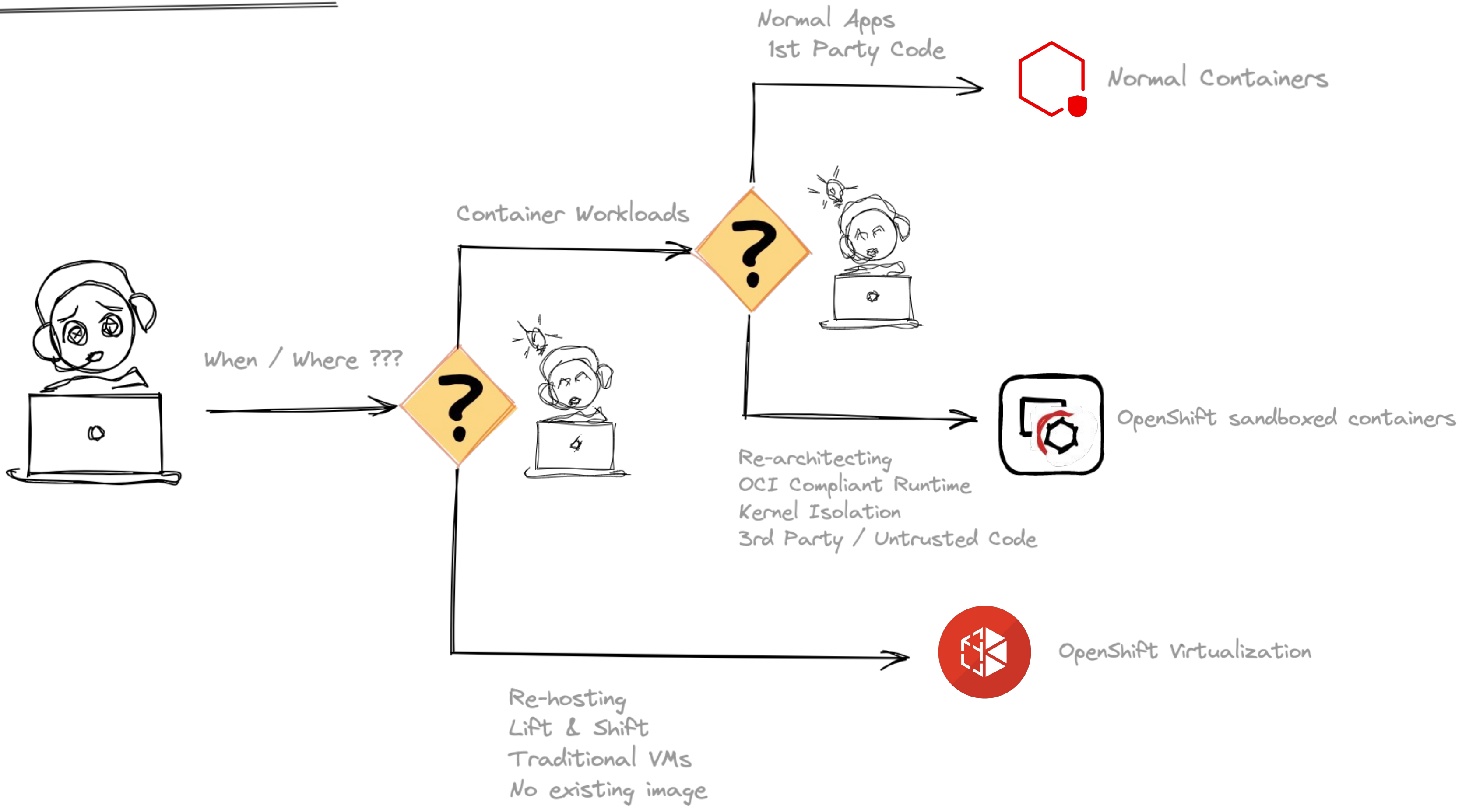
Lazy Mode (not optimal)

Opt-in only if you you know what you are doing

Vegas mode

What happens in Vegas REALLY Stays in Vegas [5]

When / Where ??



When / Where ???

Container Workloads

Normal Apps
1st Party Code

Normal Containers

Re-architecting
OCI Compliant Runtime
Kernel Isolation
3rd Party / Untrusted Code

OpenShift sandboxed containers

Re-hosting
Lift & Shift
Traditional VMs
No existing image

OpenShift Virtualization

RED HAT®
OPENSHIFT

The OpenShift Bits

Basics

OLM MCO Extensions

Operator

OLM Digest

OperatorGroup

An OperatorGroup is an OLM resource that provides rudimentary multitenant configuration to OLM installed operators.

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  annotations:
    olm.providedAPIs: KataConfig.v1.kataconfiguration.openshift.io
  name: openshift-sandboxed-containers-operator-pbzwg
  namespace: openshift-sandboxed-containers-operator
spec:
  targetNamespaces:
   - openshift-sandboxed-containers-operator
```

Subscription

A Subscription represents an intention to install an operator

Subscriptions describe which channel of an operator package to subscribe to, and whether to perform updates automatically or manually

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  labels:
    operators.coreos.com/sandboxed-containers-operator.openshift-sandboxed-containers-op: ""
  name: sandboxed-containers-operator
  namespace: openshift-sandboxed-containers-operator
spec:
  channel: preview-1.0
  installPlanApproval: Automatic
  name: sandboxed-containers-operator
  source: ....
  sourceNamespace: openshift-marketplace
  startingCSV: sandboxed-containers-operator.v1.0.0
```

ClusterServiceVersion (CSV)

A ClusterServiceVersion (CSV) represents a particular version a running operator on a cluster. It includes metadata such as name, description, version, repository link, labels, icon, etc.

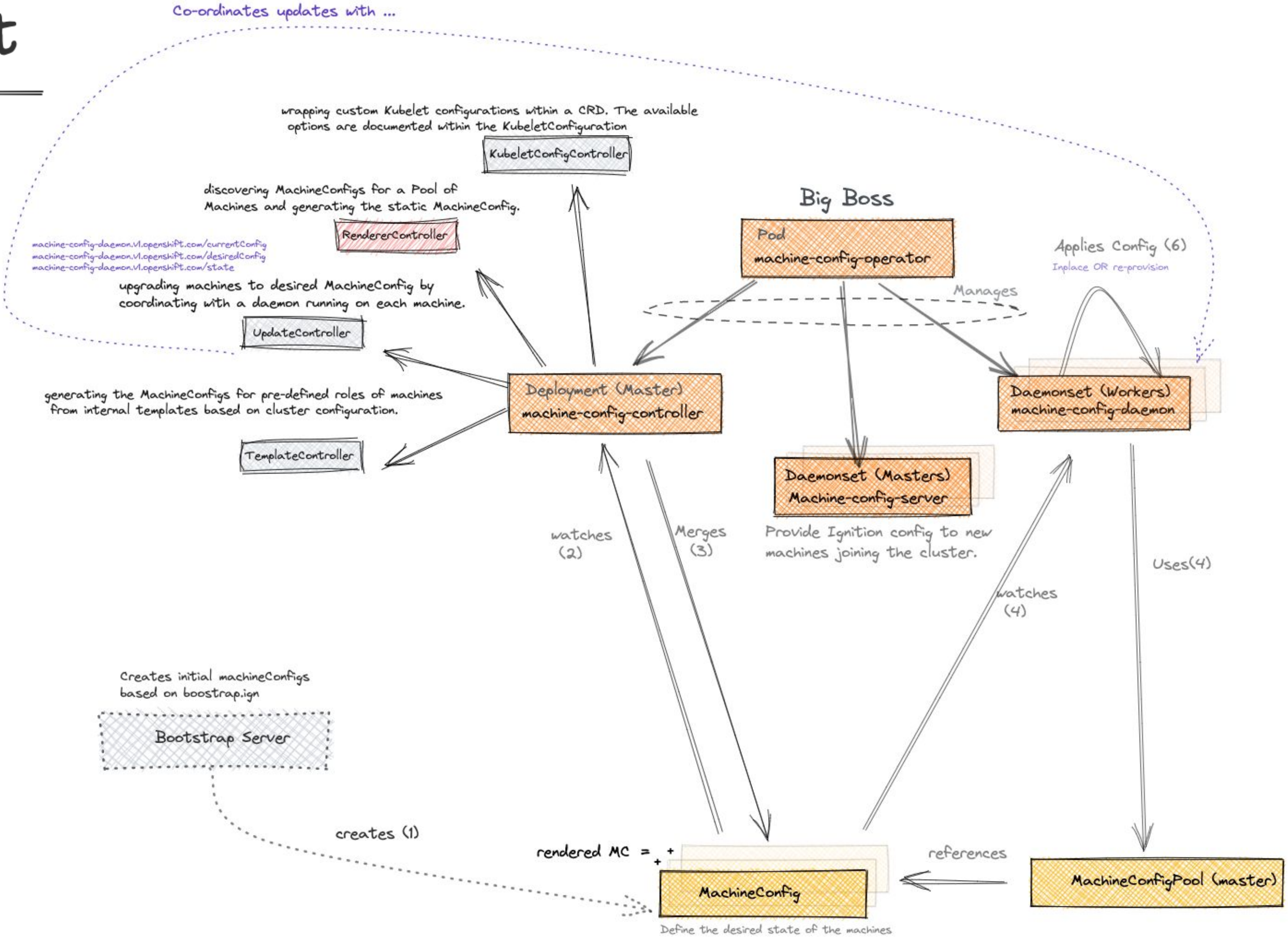
It declares owned/required CRDs, cluster requirements, and install strategy that tells OLM how to create required resources and set up the operator as a deployment.

```
apiVersion: operators.coreos.com/v1alpha1
kind: ClusterServiceVersion
metadata:
  annotations:
    alm-examples: |-
    capabilities: Basic Install
    olm.operatorGroup: openshift-sandboxed-containers-operator-pbzwg
    ....
  labels:
    operators.coreos.com/sandboxed-containers-operator.openshift-sandboxed-containers-op: ""
  name: sandboxed-containers-operator.v1.0.0
  namespace: openshift-sandboxed-containers-operator
spec:
  cleanup:
    enabled: false
  customresourcedefinitions:
    owned:
    - description: The kataconfig CR represent a installation of Kata in a cluster
      and its current state.

     kind: KataConfig
           name: kataconfigs.kataconfiguration.openshift.io

      version: v1
  description: An operator to perform lifecycle management (install/upgrade/uninstall)
    of Sandboxed Containers Runtime on Openshift as well as Kubernetes cluster
  displayName: OpenShift sandboxed containers Operator
```

MCO Digest



Co-ordinates updates with ...

wrapping custom Kubelet configurations within a CRD. The available options are documented within the KubeletConfiguration

KubeletConfigController

discovering MachineConfigs for a Pool of Machines and generating the static MachineConfig.

RendererController

machine-config-daemon.v1.openshift.com/currentConfig
machine-config-daemon.v1.openshift.com/desiredConfig
machine-config-daemon.v1.openshift.com/state

upgrading machines to desired MachineConfig by coordinating with a daemon running on each machine.

UpdateController

generating the MachineConfigs for pre-defined roles of machines from internal templates based on cluster configuration.

TemplateController

Big Boss

Pod
machine-config-operator

Manages

Applies Config (6)
Inplace OR re-provision

Deployment (Master)
machine-config-controller

Daemonset (Workers)
machine-config-daemon

Daemonset (Masters)
Machine-config-server

Provide Ignition config to new machines joining the cluster.

watches (2)

Merges (3)

watches (4)

Uses(4)

Creates initial machineConfigs based on boostrap.ign

Bootstrap Server

creates (1)

rendered MC = + +

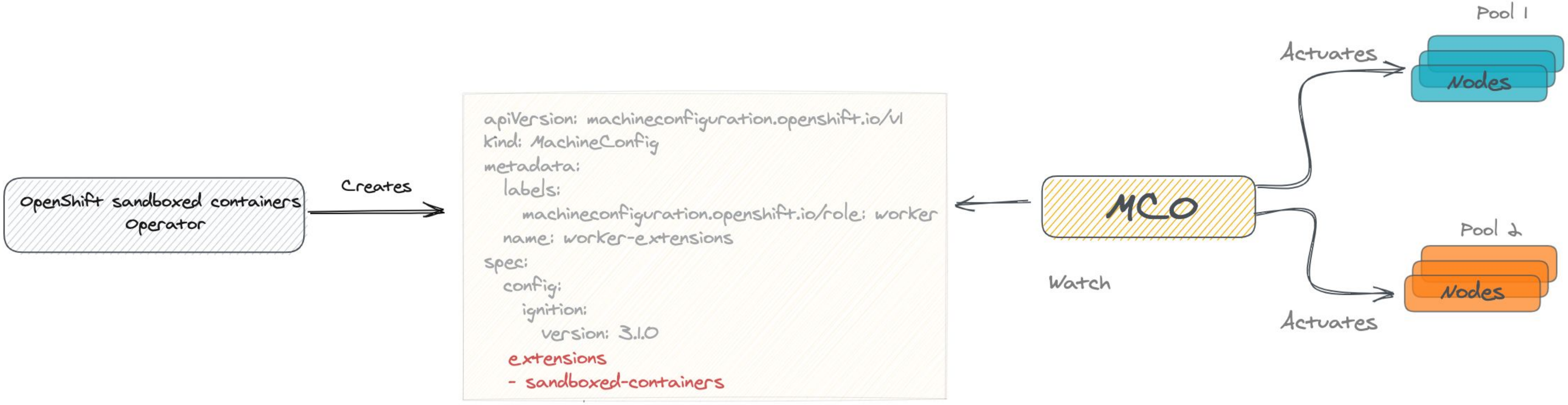
MachineConfig

Define the desired state of the machines

references

MachineConfigPool (master)

RHCOS Extensions



```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: worker-extensions
spec:
  config:
    ignition:
      version: 3.1.0
  extensions
  - sandboxed-containers
```

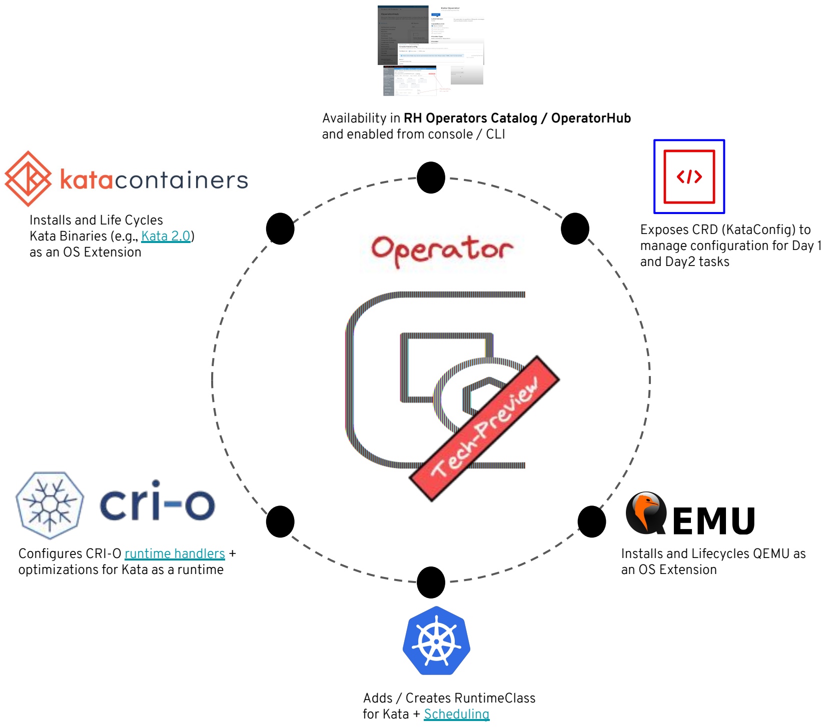
This translates to a list of packages

- kata-containers
- qemu-kiwi
- ipxe-roms-qemu
- libpmem
- pixman
- qemu-kiwi
- qemu-kvm-common
- seabios-bin
- seavgabios-bin
- sgabios-bin

The Operator

Tech-Preview

Kata Containers as as Service (Operator machinery)



Availability in **RH Operators Catalog / OperatorHub** and enabled from console / CLI

Exposes CRD (KataConfig) to manage configuration for Day 1 and Day2 tasks

Installs and Lifecycles QEMU as an OS Extension

Adds / Creates RuntimeClass for Kata + Scheduling

Configures CRI-O runtime handlers + optimizations for Kata as a runtime

Installs and Life Cycles Kata Binaries (e.g., Kata 2.0) as an OS Extension

Usage Manual

Admin creates KataConfig (optionally selects nodes that will have the Kata runtime enabled)

Cluster Admin

```
apiVersion: kataconfiguration.openshift.io/v1
kind: KataConfig
metadata:
 name: example-kataconfig
spec:
 kataConfigPoolSelector:
   matchLabels:
     node-label-kata: test
```

Operator automagically enables Kata on the nodes and creates the RuntimeClass

The Operator

```
apiVersion: node.k8s.io/v2
kind: RuntimeClass
metadata:
 name: my-kata-class
Handler: kata
```

Developer

Developers defines the RuntimeClass at the Deployment / Pod level to use Kata

```
apiVersion: v1
kind: Pod
metadata:
 name: mypod
spec:
  runtimeClassName: kata
```

Its DEMO Time!

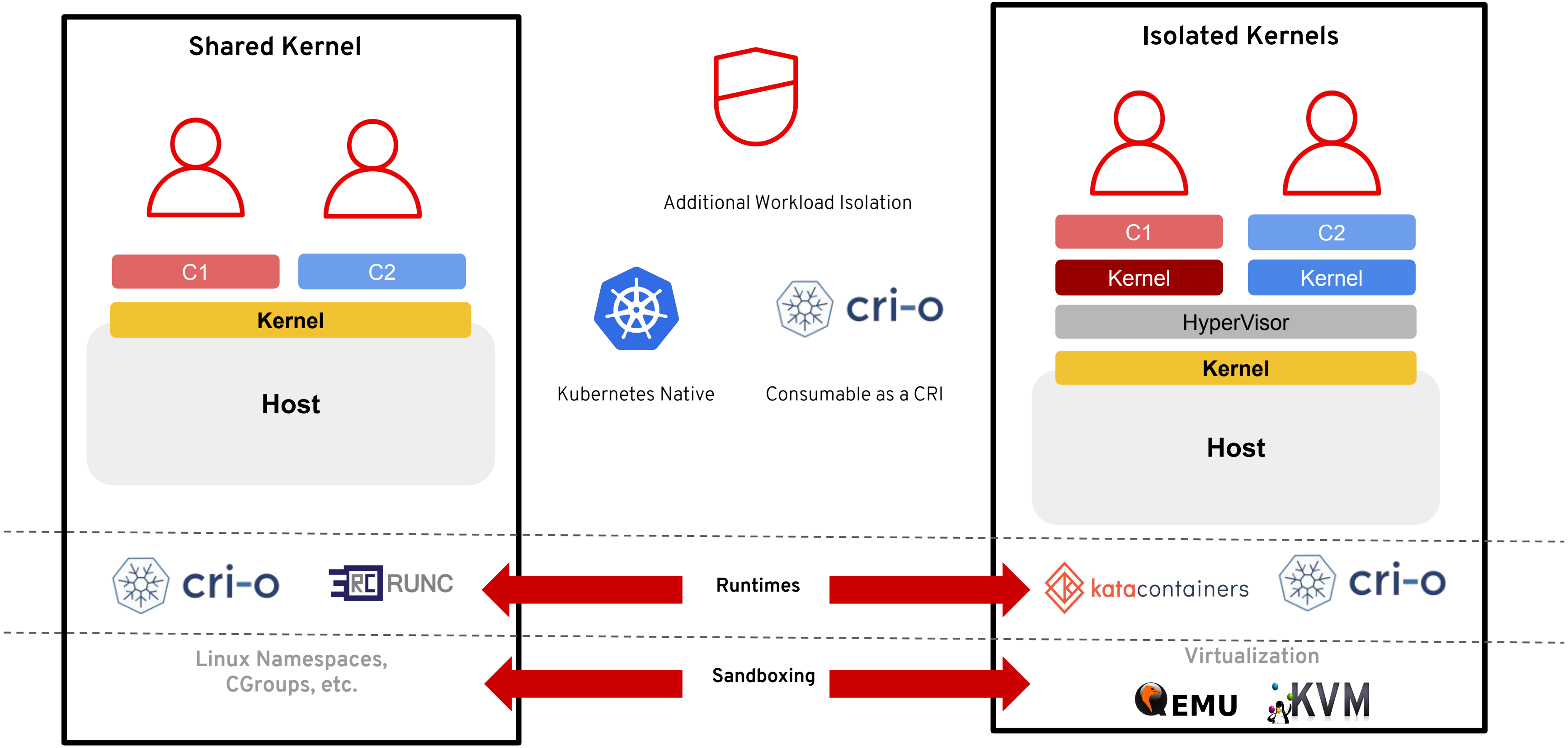
RED HAT®
OPENSHIFT

Kata Containers: Deep-Dive

High-level Arch Vs. Our Stack

E2E Flow

High-level Arch Vs.



Shared Kernel

C1

C2

Kernel

Host

cri-o

RUNC

Linux Namespaces, CGroups, etc.

Additional Workload Isolation

Kubernetes Native

cri-o

Consumable as a CRI

Runtimes

Sandboxing

Isolated Kernels

C1

C2

Kernel

Kernel

HyperVisor

Kernel

Host

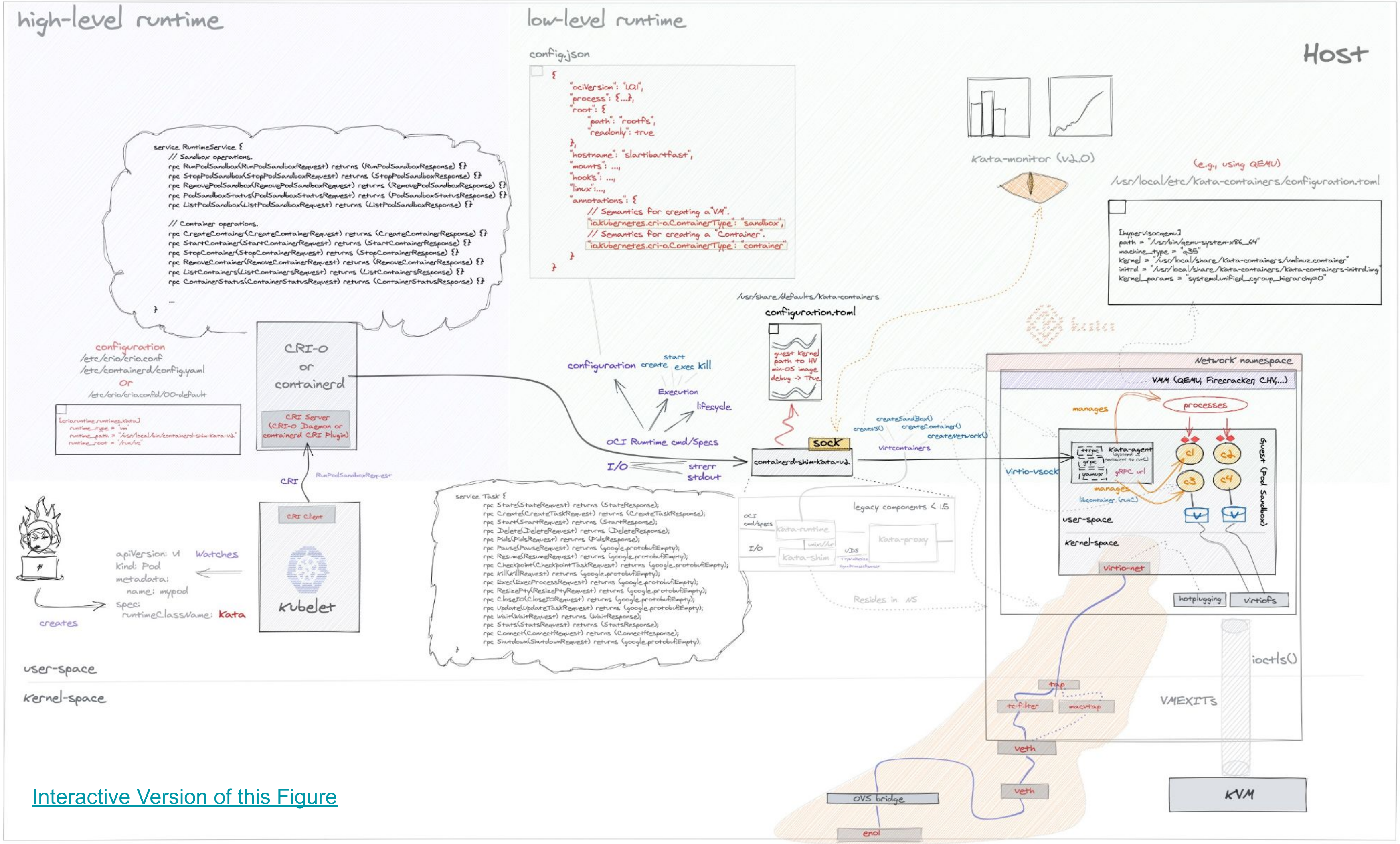
katacontainers

cri-o

Virtualization

QEMU

KVM



```
service RuntimeService {
    // Sandbox operations.
    rpc RunPodSandbox(RunPodSandboxRequest) returns (RunPodSandboxResponse) {}
    rpc StopPodSandbox(StopPodSandboxRequest) returns (StopPodSandboxResponse) {}
    rpc RemovePodSandbox(RemovePodSandboxRequest) returns (RemovePodSandboxResponse) {}
    rpc PodSandboxStatus(PodSandboxStatusRequest) returns (PodSandboxStatusResponse) {}
    rpc ListPodSandbox(ListPodSandboxRequest) returns (ListPodSandboxResponse) {}

    // Container operations.
    rpc CreateContainer(CreateContainerRequest) returns (CreateContainerResponse) {}
    rpc StartContainer(StartContainerRequest) returns (StartContainerResponse) {}
    rpc StopContainer(StopContainerRequest) returns (StopContainerResponse) {}
    rpc RemoveContainer(RemoveContainerRequest) returns (RemoveContainerResponse) {}
    rpc ListContainers(ListContainersRequest) returns (ListContainersResponse) {}
    rpc ContainerStatus(ContainerStatusRequest) returns (ContainerStatusResponse) {}

    ...
}
```

```
{
    "ociVersion": "1.0.1",
    "process": {...},
    "root": {
        "path": "rootfs",
        "readonly": true
    },
    "hostname": "slartibartfast",
    "mounts": ...,
    "hooks": ...,
    "linux":...,
    "annotations": {
        // Semantics for creating a "VM".
        "io.kubernetes.cri-o.ContainerType": "sandbox",
        // Semantics for creating a "Container".
        "io.kubernetes.cri-o.ContainerType": "container"
    }
}
```

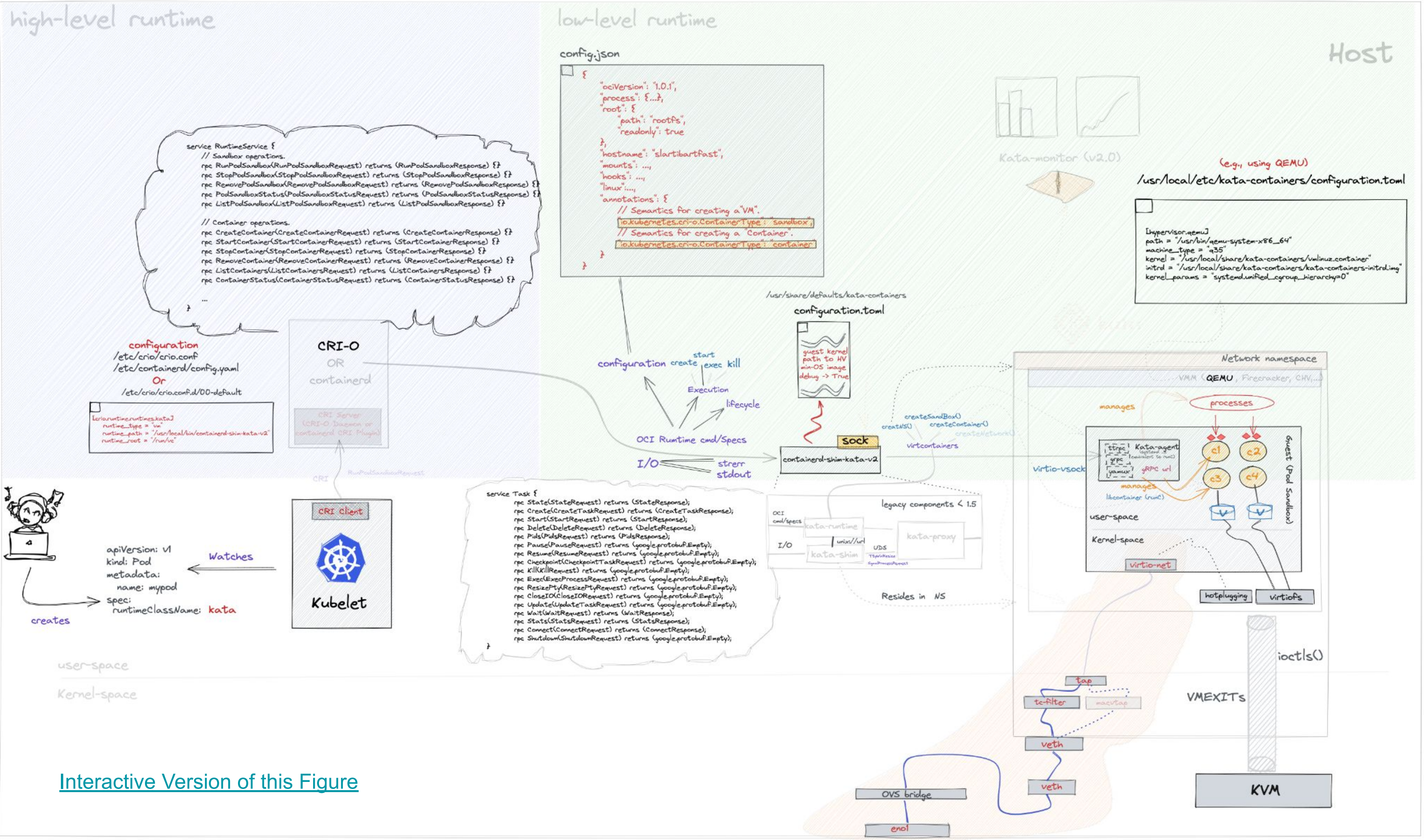
```
[hypervisor.qemu]
path = "/usr/bin/qemu-system-x86_64"
machine_type = "q35"
kernel = "/usr/local/share/kata-containers/vmlinuz.container"
initrd = "/usr/local/share/kata-containers/kata-containers-initrd.img"
kernel_params = "systemd.unified_cgroup_hierarchy=0"
```

```
[crio.runtime.runtimes.kata]
    runtime_type = "vm"
    runtime_path = "/usr/local/bin/containerd-shim-kata-v2"
    runtime_root = "/run/vc"
```

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  runtimeClassName: kata
```

```
service Task {
        rpc State(StateRequest) returns (StateResponse);
        rpc Create(CreateTaskRequest) returns (CreateTaskResponse);
        rpc Start(StartRequest) returns (StartResponse);
        rpc Delete(DeleteRequest) returns (DeleteResponse);
        rpc Pids(PidsRequest) returns (PidsResponse);
        rpc Pause(PauseRequest) returns (google.protobuf.Empty);
        rpc Resume(ResumeRequest) returns (google.protobuf.Empty);
        rpc Checkpoint(CheckpointTaskRequest) returns (google.protobuf.Empty);
        rpc Kill(KillRequest) returns (google.protobuf.Empty);
        rpc Exec(ExecProcessRequest) returns (google.protobuf.Empty);
        rpc ResizePty(ResizePtyRequest) returns (google.protobuf.Empty);
        rpc CloseIO(CloseIORequest) returns (google.protobuf.Empty);
        rpc Update(UpdateTaskRequest) returns (google.protobuf.Empty);
        rpc Wait(WaitRequest) returns (WaitResponse);
        rpc Stats(StatsRequest) returns (StatsResponse);
        rpc Connect(ConnectRequest) returns (ConnectResponse);
        rpc Shutdown(ShutdownRequest) returns (google.protobuf.Empty);
}
```

Interactive Version of this Figure



```
service RuntimeService {
    // Sandbox operations.
    rpc RunPodSandbox(RunPodSandboxRequest) returns (RunPodSandboxResponse) {}
    rpc StopPodSandbox(StopPodSandboxRequest) returns (StopPodSandboxResponse) {}
    rpc RemovePodSandbox(RemovePodSandboxRequest) returns (RemovePodSandboxResponse) {}
    rpc PodSandboxStatus(PodSandboxStatusRequest) returns (PodSandboxStatusResponse) {}
    rpc ListPodSandbox(ListPodSandboxRequest) returns (ListPodSandboxResponse) {}

    // Container operations.
    rpc CreateContainer(CreateContainerRequest) returns (CreateContainerResponse) {}
    rpc StartContainer(StartContainerRequest) returns (StartContainerResponse) {}
    rpc StopContainer(StopContainerRequest) returns (StopContainerResponse) {}
    rpc RemoveContainer(RemoveContainerRequest) returns (RemoveContainerResponse) {}
    rpc ListContainers(ListContainersRequest) returns (ListContainersResponse) {}
    rpc ContainerStatus(ContainerStatusRequest) returns (ContainerStatusResponse) {}

    ...
}
```

configuration
/etc/crio/crio.conf
/etc/containerd/config.yaml
Or
/etc/crio/crio.conf.d/00-default

```
[crio.runtime.runtimes.kata]
    runtime_type = "vm"
    runtime_path = "/usr/local/bin/containerd-shim-kata-v2"
    runtime_root = "/run/vc"
```

config.json

```
{
    "ociVersion": "1.0.1",
    "process": {...},
    "root": {
        "path": "rootfs",
        "readonly": true
    },
    "hostname": "slartibartfast",
    "mounts": ...,
    "hooks": ...,
    "linux":...,
    "annotations": {
        // Semantics for creating a"VM".
        "io.kubernetes.cri-o.ContainerType": "sandbox",
        // Semantics for creating a "Container".
        "io.kubernetes.cri-o.ContainerType": "container"
    }
}
```

(e.g., using QEMU)
/usr/local/etc/kata-containers/configuration.toml

```
[hypervisor.qemu]
path = "/usr/bin/qemu-system-x86_64"
machine_type = "q35"
kernel = "/usr/local/share/kata-containers/vmlinuz.container"
initrd = "/usr/local/share/kata-containers/kata-containers-initrd.img"
kernel_params = "systemd.unified_cgroup_hierarchy=0"
```

/usr/share/defaults/kata-containers
configuration.toml

guest kernel
path to HV
min-OS image
debug -> True

```
service Task {
        rpc State(StateRequest) returns (StateResponse);
        rpc Create(CreateTaskRequest) returns (CreateTaskResponse);
        rpc Start(StartRequest) returns (StartResponse);
        rpc Delete(DeleteRequest) returns (DeleteResponse);
        rpc Pids(PidsRequest) returns (PidsResponse);
        rpc Pause(PauseRequest) returns (google.protobuf.Empty);
        rpc Resume(ResumeRequest) returns (google.protobuf.Empty);
        rpc Checkpoint(CheckpointTaskRequest) returns (google.protobuf.Empty);
        rpc Kill(KillRequest) returns (google.protobuf.Empty);
        rpc Exec(ExecProcessRequest) returns (google.protobuf.Empty);
        rpc ResizePty(ResizePtyRequest) returns (google.protobuf.Empty);
        rpc CloseIO(CloseIORequest) returns (google.protobuf.Empty);
        rpc Update(UpdateTaskRequest) returns (google.protobuf.Empty);
        rpc Wait(WaitRequest) returns (WaitResponse);
        rpc Stats(StatsRequest) returns (StatsResponse);
        rpc Connect(ConnectRequest) returns (ConnectResponse);
        rpc Shutdown(ShutdownRequest) returns (google.protobuf.Empty);
}
```

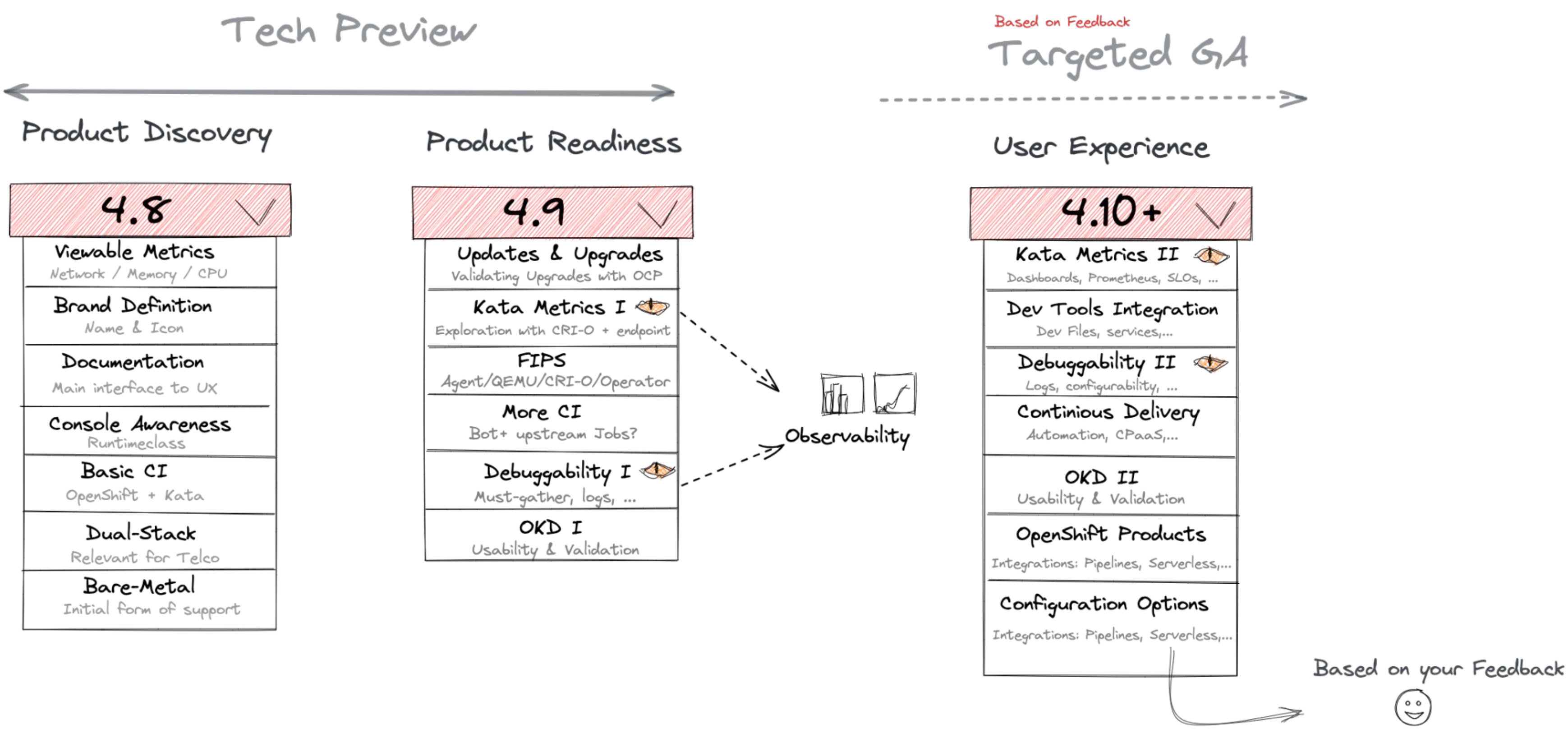
```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  runtimeClassName: kata
```

Interactive Version of this Figure

Its DEMO Time!

RED HAT®
OPENSHIFT

The Pipeline (Roadmap)



Tech Preview

Based on Feedback

Targeted GA

Product Discovery

4.8

- Viewable Metrics — Network / Memory / CPU
- Brand Definition — Name & Icon
- Documentation — Main interface to UX
- Console Awareness — Runtimeclass
- Basic CI — OpenShift + Kata
- Dual-Stack — Relevant for Telco
- Bare-Metal — Initial form of support

Product Readiness

4.9

- Updates & Upgrades — Validating Upgrades with OCP
- Kata Metrics I — Exploration with CRI-O + endpoint
- FIPS — Agent/QEMU/CRI-O/Operator
- More CI — Bot+ upstream Jobs?
- Debuggability I — Must-gather, logs, ...
- OKD I — Usability & Validation

Observability

User Experience

4.10+

- Kata Metrics II — Dashboards, Prometheus, SLOs, ...
- Dev Tools Integration — Dev Files, services,...
- Debuggability II — Logs, configurability, ...
- Continious Delivery — Automation, CPaaS,...
- OKD II — Usability & Validation
- OpenShift Products — Integrations: Pipelines, Serverless,...
- Configuration Options — Integrations: Pipelines, Serverless,...

Based on your Feedback

Disclaimer: Subject To Change

RED HAT®
OPENSHIFT

References

[1] Practical and effective sandboxing for Linux containers

[2] User-level Resource-constrained Sandboxing

[3] Sandbox - Wikipedia

[4] Jain, Madhur. "Study of Firecracker MicroVM."

[5] SELinux changes for KVM-separated (Kata) containers

[6] Interactive Version of Kata Containers