KubeCon | CloudNativeCon

North America 2022

BUILDING FOR THE ROAD AHEAD

# DETROIT 2022

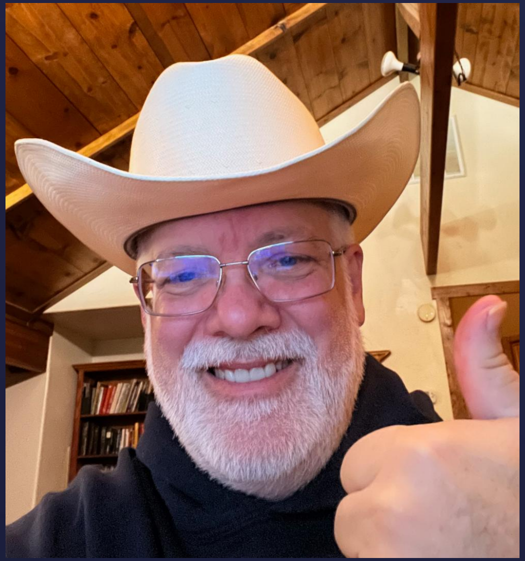
# NRI: Extending Containerd and CRI-O With Common Plugins

KubeCon | CloudNativeCon
North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

**October 24-28, 2021**



**Mike Brown**
OSS Software Engineer
*IBM*

**Krisztian Litkey**
Software Engineer
*Intel Finland*

# We’ll Cover…

- Some History And Background
- What Is NRI ?
- How Does It Work ?
- What Can It Do ?
- Writing An NRI Plugin ?
- Examples, Use Cases
- Q & A

# History And Background

- various extension mechanisms over the years
- OCI hooks
  - custom actions executed at various container lifecycle events
  - runc, cri-o
- runtime wrapper using runtime classes
  - custom actions implemented by (custom) runtime wrapper
  - a bit of a kludge
  - runc/crun/etc…, containerd/cri-o
- NRI
  - custom actions at pod or container creation/stop
  - containerd
- NRI evolution
  - subject of this presentation…

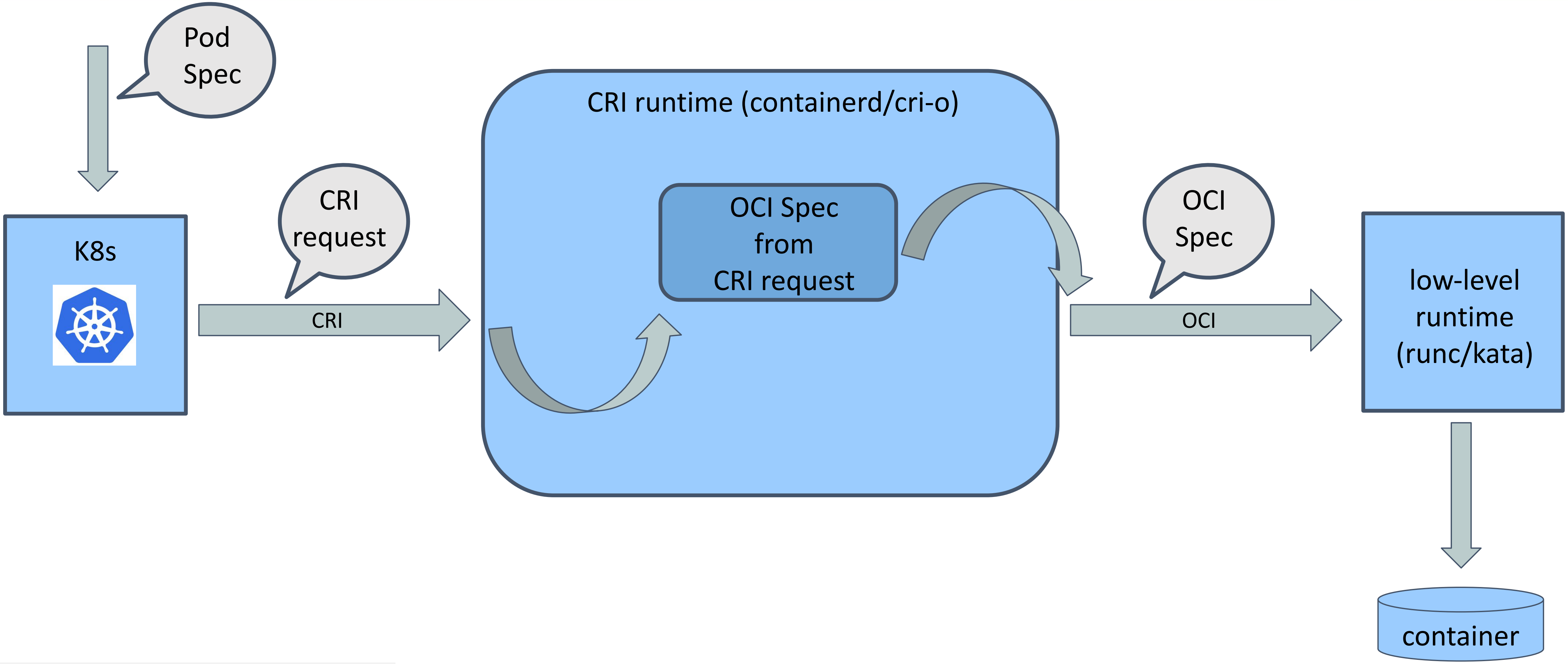
# What Is NRI ?

- NRI: Node Resource Interface, a common framework for
  - plugging extensions into OCI-compatible runtimes
  - implementing custom container configuration logic
- ‘common’
  - plugins work identically in supported runtimes
  - support present in commonly used runtimes
- ‘framework’
  - multiple NRI pieces
  - collectively achieve NRI’s goal
- ‘pluggable extensions’
  - alter behavior without runtime modifications
- ‘custom logic’
  - your cluster, your plugin, your rules
  - within the allowed boundaries imposed by NRI

# What Is NRI…

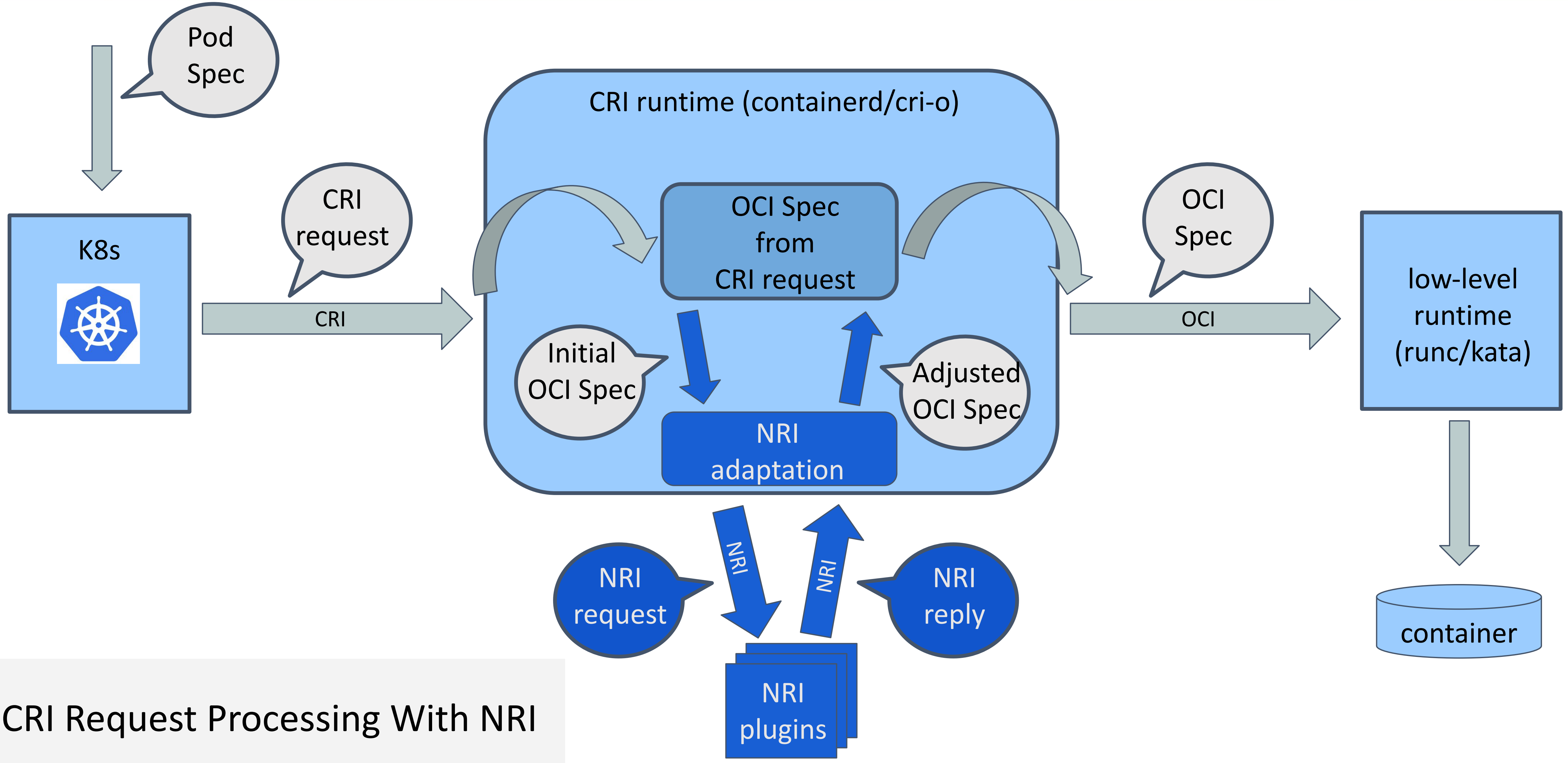
- plumbing to enforce how containers are configured
  - initially, during container creation
  - subsequently, by container updates
  - potentially, in response to some external events
- => your cluster, your plugin, your rules
  - within the allowed boundaries imposed by NRI

# How Does NRI Work ?



CRI Request Processing

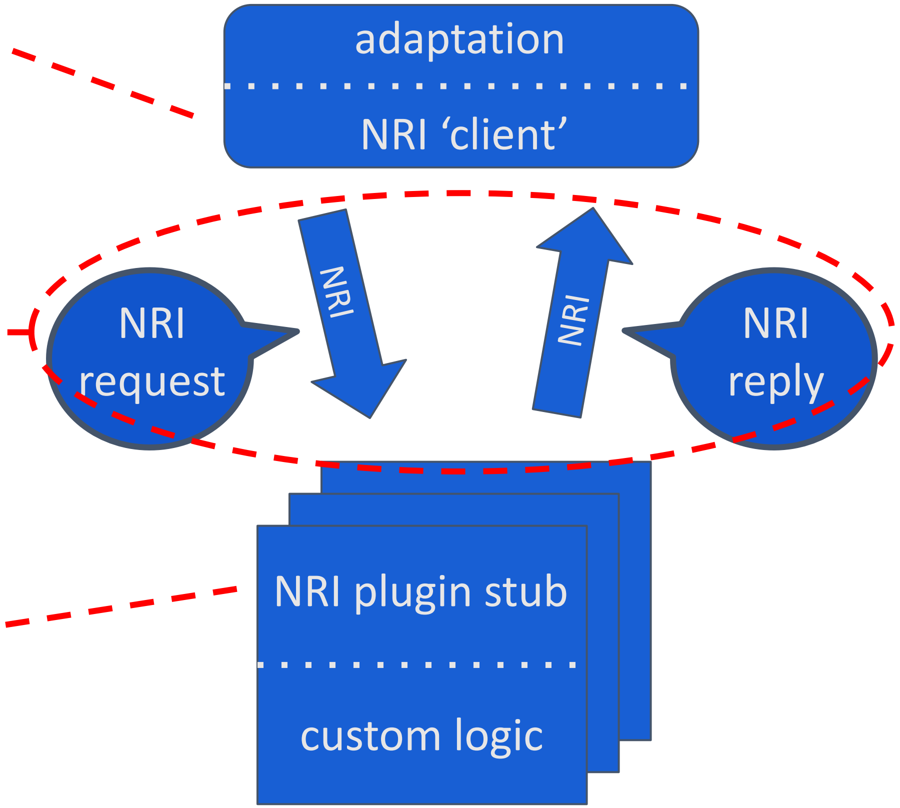
# How Does NRI Work...



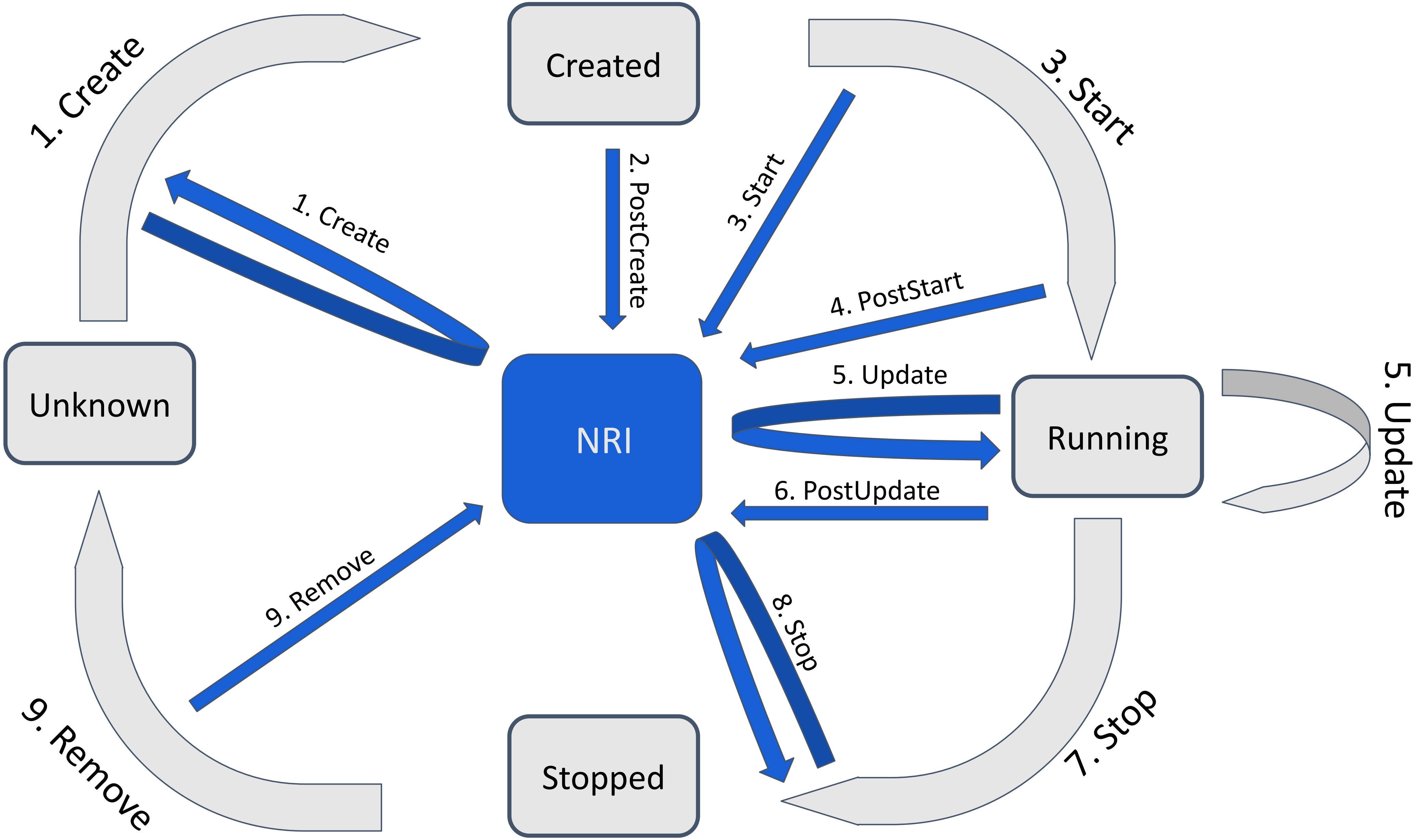
CRI Request Processing With NRI

# How Does NRI Work…

- runtime adaptation client
  - runtime agnostic
  - integrate to NRI / interact with plugins
- NRI protocol
  - protobuf-based 'API'
  - ttRPC bindings
  - execution model
    - full set of events/request
  - data model: event/request details
    - pods, containers (subset of OCI Spec)
    - NRI container adjustments, updates
- plugin stub
  - low-level details of plugin logic
  - IPC, connection, registration, etc.



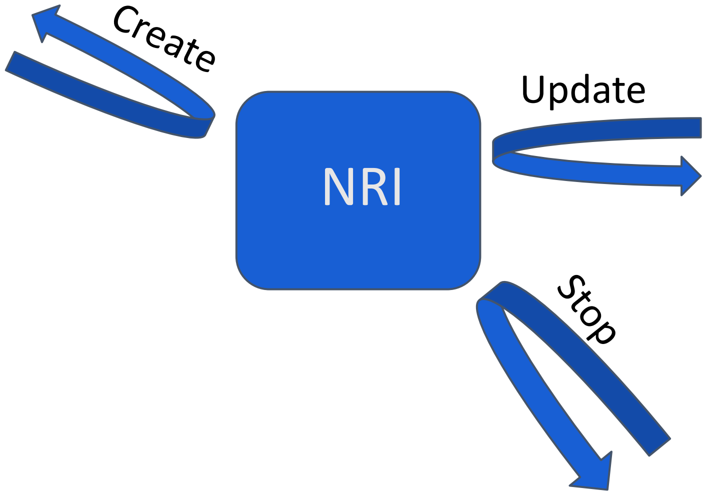
# NRI Plugin Events



# NRI Plugin Input And Output

container creation response:

- container being created
  - annotations, env. vars.
  - mounts, devices
  - OCI hooks
  - ‘resources’
    - scheduling params
    - CPU/memory pinning
    - memory, huge page limits
    - LLC cache allocation
    - blkio controller parameters
- existing containers
  - ‘resources’



container update response:

- container being updated
  - ‘resources’
- other containers
  - ‘resources’

container stop response:

- stopped container
  - nothing
- remaining containers
  - ‘resources’

container request includes:

- labels, annotations, command, env. vars.
- mounts, devices, OCI hooks
- ‘resources’
- runtime handler, cgroup path

# What Else NRI Plugins Can Do…

- plugins can also
  - synchronize state on connection establishment
  - make unsolicited changes to containers
  - subscribe and react to Pod lifecycle events
  - control other things, outside the scope of the OCI Spec
- plugins can’t
  - make conflicting changes to the same container
  - NRI catches this and treats it as an error
- plugins shouldn’t
  - try to manipulate directly anything covered by the OCI Spec
  - even if those things are not covered/controllable via NRI

# How To Write An NRI Plugin

- clone the NRI plugin template, fill in the missing details
  - implement data collection
    - if you need to collect pod/container data subscribe for the corresponding events
    - if you need data external to NRI/OCI, implement collection (eg. metrics)
  - implement custom container setup
    - subscribe for container creation requests
    - provide container customization response
  - implement custom container update, if needed
    - subscribe for container update, stop requests
    - provide update for other containers in create response
    - provide update for containers in update, stop responses
- proactive custom container configuration logic
  - collect any data/metrics you need for updating container configuration
  - trigger ‘calculation’ of new configurations
  - set new configurations using an unsolicited update request
- and… look at our sample plugins for simple examples

# Examples And Use Cases

- ‘real world’ plugins
  - annotation-based device injection
  - CDI device injection (obsolete)
  - OCI hook injection
- debugging and development plugins
  - event logger
  - container differ
  - plugin template
- experimental resource assignment plugin
  - CRI Resource Manager with ‘--use-nri-plugin’

# Thank You ! / Q & A

Contact info:

- Krisztian Litkey, krisztian.litkey@intel.com, klihub on github
- Mike Brown, brownwm@us.ibm.com, mikebrow on github

NRI Artifacts:

- https://github.com/containerd/nri/pull/16
- https://github.com/containerd/containerd/pull/6019
- https://github.com/cri-o/cri-o/pull/5318

- any questions or comments ?

Please scan the QR Code above to leave feedback on this session