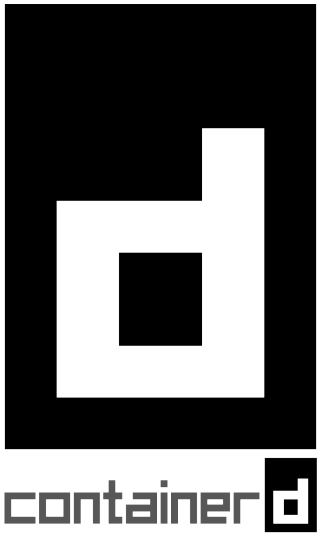
NRI Intro

An Introduction to NRI, Use Cases and Demos

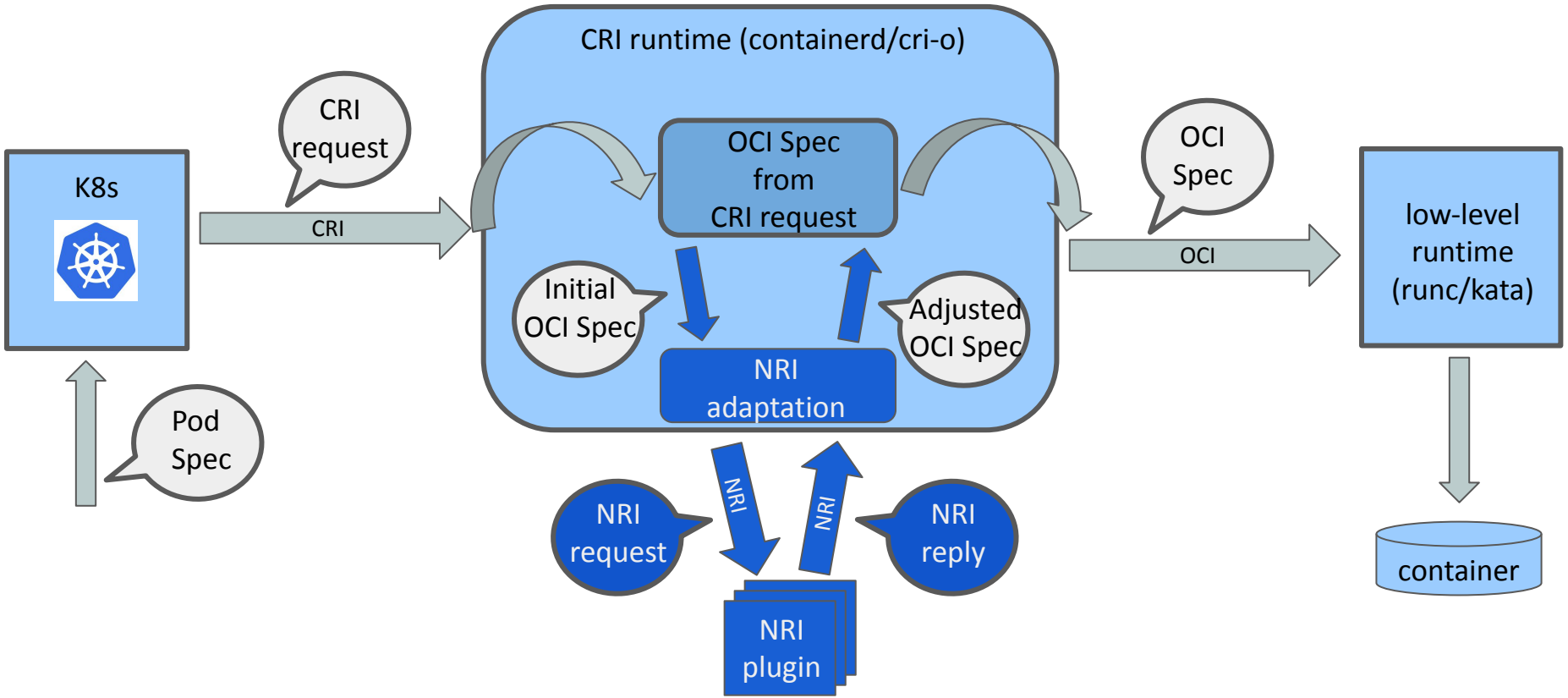
Krisztian Litkey, Peter Hunt

What is NRI ?

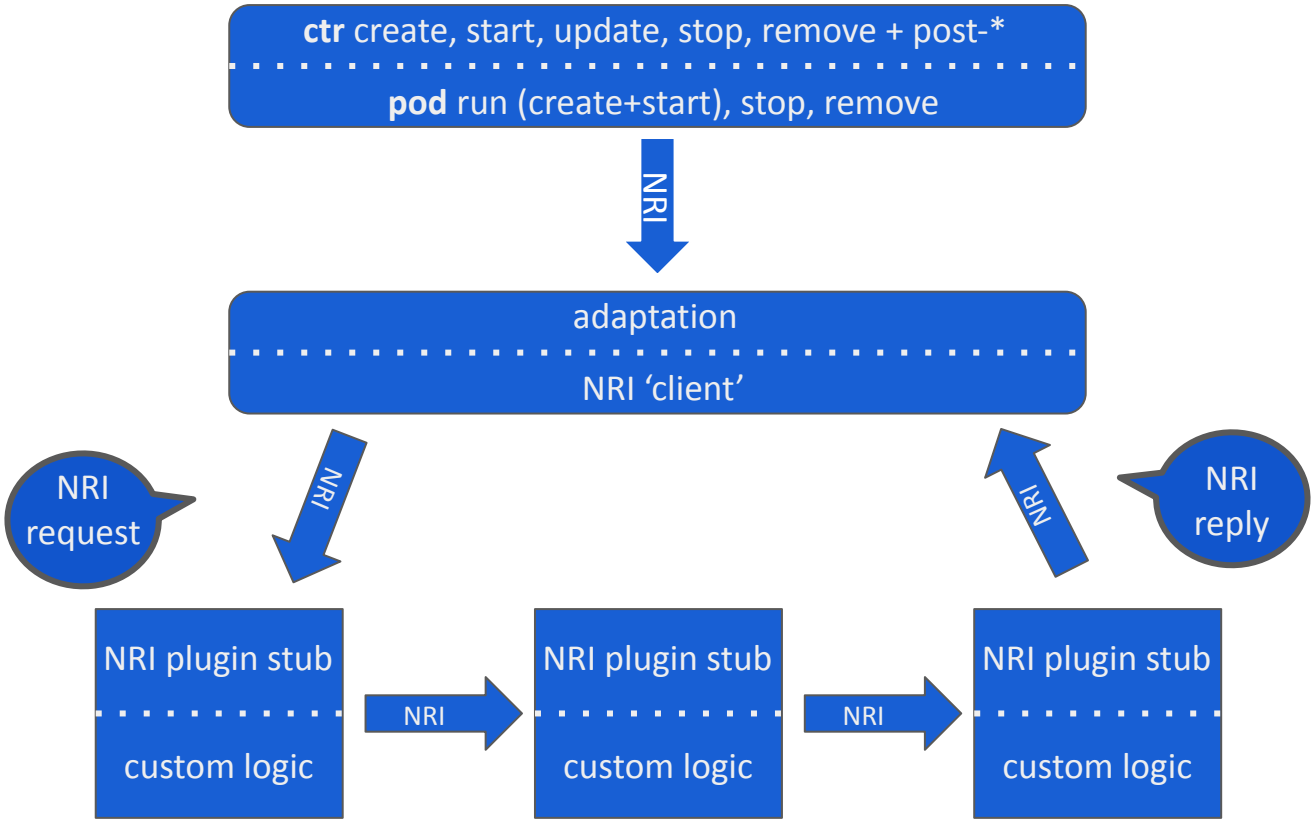
Common interface and framework for generic modifications to OCI specifications for CRI-compatible container managers



How Does NRI Work ?

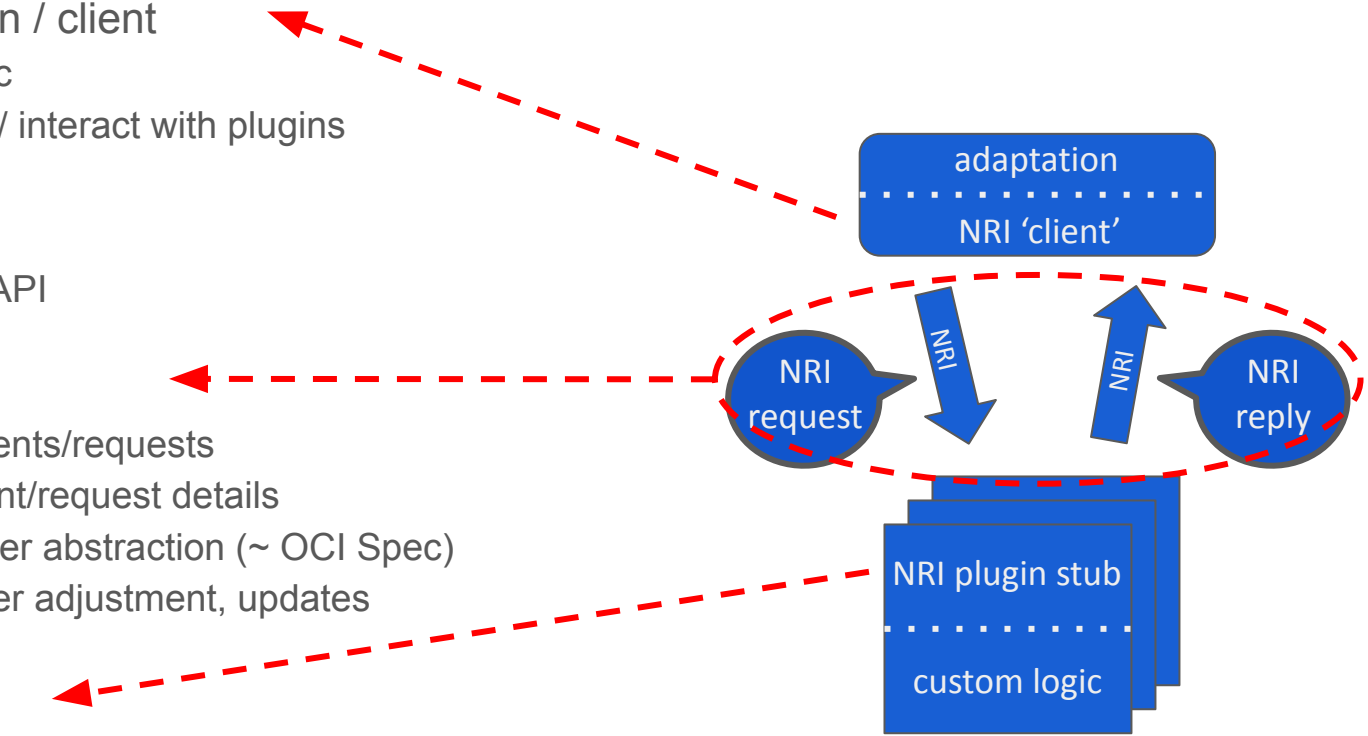


How Does NRI Work ?



How Does NRI Work ?

- Runtime adaptation / client
  - Runtime agnostic
  - Integrate to NRI / interact with plugins
- NRI protocol
  - Protobuf-based API
  - ttRPC bindings
  - Execution model
    - Lifecycle events/requests
  - Data model: event/request details
    - pod, container abstraction (~ OCI Spec)
    - NRI container adjustment, updates
- Plugin stub
  - Common low-level plugin logic
  - IPC, connection, registration, etc.



How Does NRI Work ?

- NRI a ‘framework’, because it has multiple pieces
- Runtime adaptation client
  - Runtime agnostic library
  - Used to integrate runtimes to NRI and to interact with plugins
- Plugin stub
  - Plugin library, takes care of common plugin tasks
  - Connection setup and communication, plugin registration, NRI event subscription, etc.
- NRI protocol
  - A protobuf-based API definition with ttRPC bindings
  - Defines the data NRI associates with pods and containers
  - Defines the lifecycle events NRI provides for pods and containers
  - Defines other events/requests NRI provides for plugins (registration, unsolicited updates)

What Can I Do With NRI ? - Events/Requests

- NRI allows container customization in response to 3 lifecycle events
  - CreateContainer, UpdateContainerResources, and StopContainer
- CreateContainer response
  - Allows the widest range of changes for the container being created: ‘container adjustment’
  - Allows existing containers’ ‘resources’ to be updated: ‘container update’
- UpdateContainerResources response
  - Allows altering the ‘resource’ update: ‘container update’
  - Allows other containers’ ‘resources’ to be updated: ‘container update’
- StopContainer response
  - Allows remaining containers’ ‘resources’ to be updated: ‘container update’
- Additionally, during plugin synchronization, or by unsolicited update requests
  - Allows containers’ ‘resources’ to be updated: ‘container update’

What Can I Do With NRI ? - Inputs

- Pod abstraction
  - K8s metadata: id, name, uid, namespace, labels, annotations
  - Linux namespaces, cgroupfs path, runtime handler name
  - Pod ‘resources’ and ‘resource’ overhead
- Container abstraction
  - Metadata: id, pod id, name, labels, annotations
  - Linux namespaces, cgroupfs path, devices
  - Command/arguments, environment variables, mounts, OCI hooks
  - ‘Resources’

What Can I Do With NRI ? - Outputs/Controls

- Container adjustment (during container creation)
  - Annotations
  - Environment variables, mounts, devices
  - OCI hooks
  - ‘Resources’
- Container update (once container is created)
  - ‘Resources’ (parameters controlled via cgroupfs)
    - Scheduling parameters
    - CPU and memory pinning
    - Memory, huge page limits
    - LLC cache allocation, block I/O throttling
    - etc.

Usage / Use Cases / Demos

- Enabling NRI in the runtime
- Inspecting NRI events
- Plugging OCI Hooks into containerd
- Sometimes you need a hack: annotated device injection
- Complex Plugin Use Case: pluggable resource policy in the runtime

Pluggable Resource Policy In the Runtime

- 2 reference policies implemented + a wireframe/template
  - Balloons: user-configured set of custom (potentially) workload-specific pools
  - Topology-aware: zero-configuration automatic resource alignment with workload isolation
- Deployment similar to device plugins: kubernetes daemonset
- Configuration using ConfigMaps
- Exposes node resource availability as NodeResourceTopology CRD’s
- Provides higher-level abstraction for resource handling on top of NRI
  - Instead of {Create,Start,Stop,}Container, you have
  - AllocateResources, UpdateResources, ReleaseResources
- Template policy: a wireframe for implementing additional policies

Current & Future Work

- Addressing the experimental bits
  - Bug fixes, better test coverage
  - Protocol completeness: any missing events, inputs, controls ?
  - Protocol correctness: proper error handling in all corner cases ?
  - Documentation
  - Other improvements
    - Less protocol stutter over the wire
    - Possible to share more NRI integration code between the runtimes?
  - Potential additions
    - CNI/networking NRI hooks ? ImageService NRI hooks ?
- Community maintained NRI plugin collection

Thank You - Q&A

- Any questions ?
- Comments ?

- https://github.com/klihub/nri-intro
  - Instructions to reproduce the demos
  - A copy of this presentation