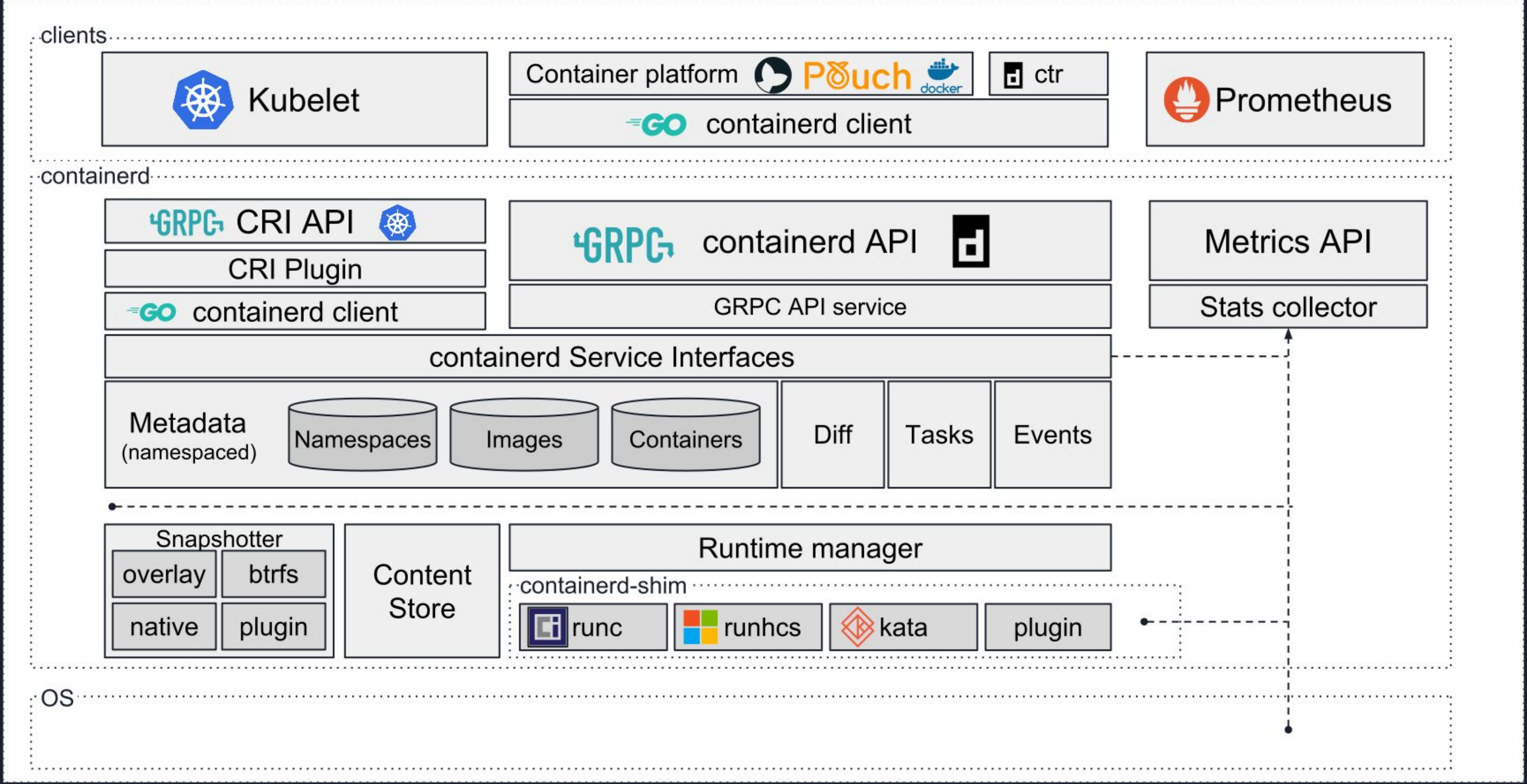
containerd deep dive

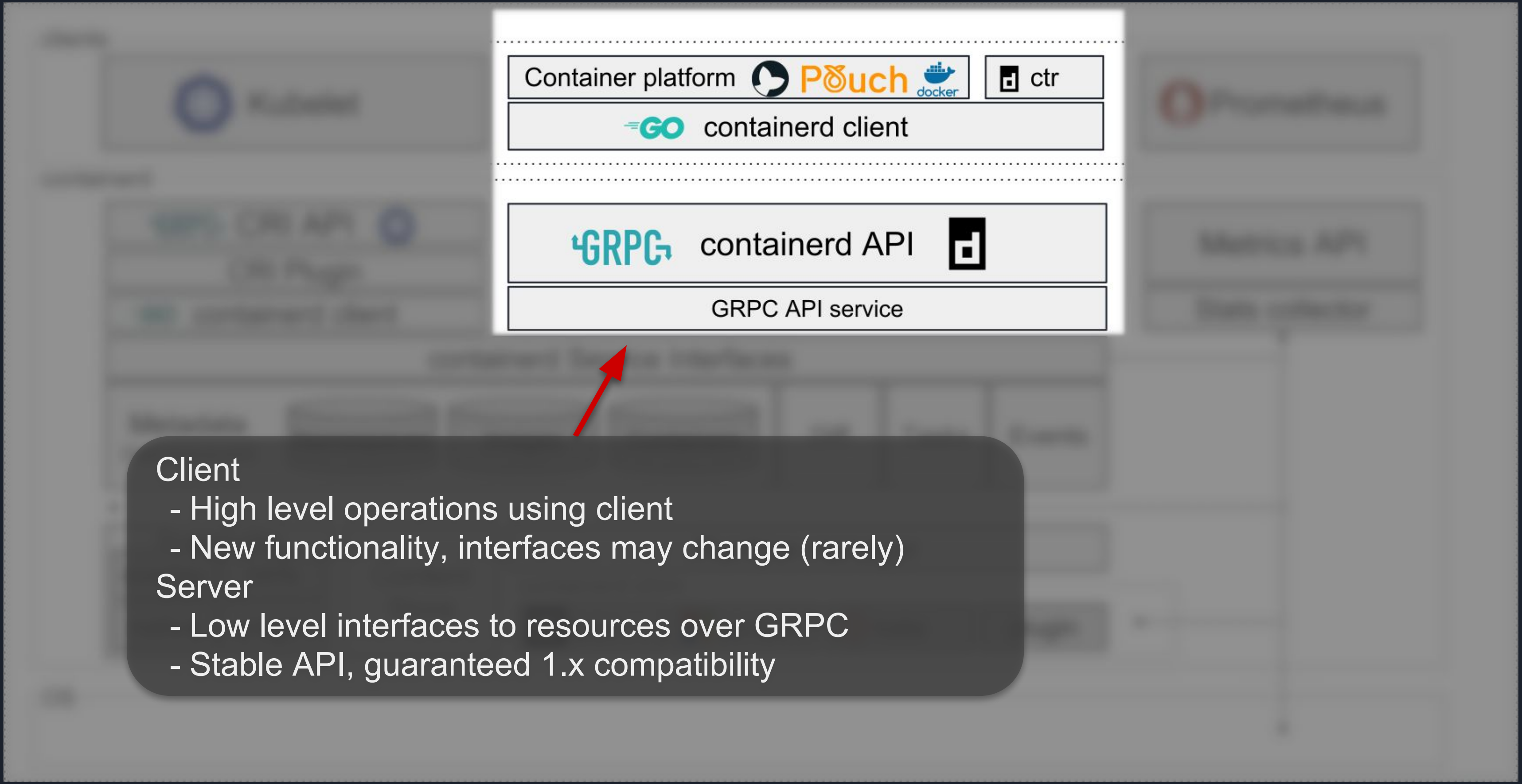
Kubecon Shanghai 2018

Derek McGowan (Docker)
Mike Brown (IBM)

Containerd 1.2 Architecture



Client-Server



Container platform

Pouch

docker

ctr

GO containerd client

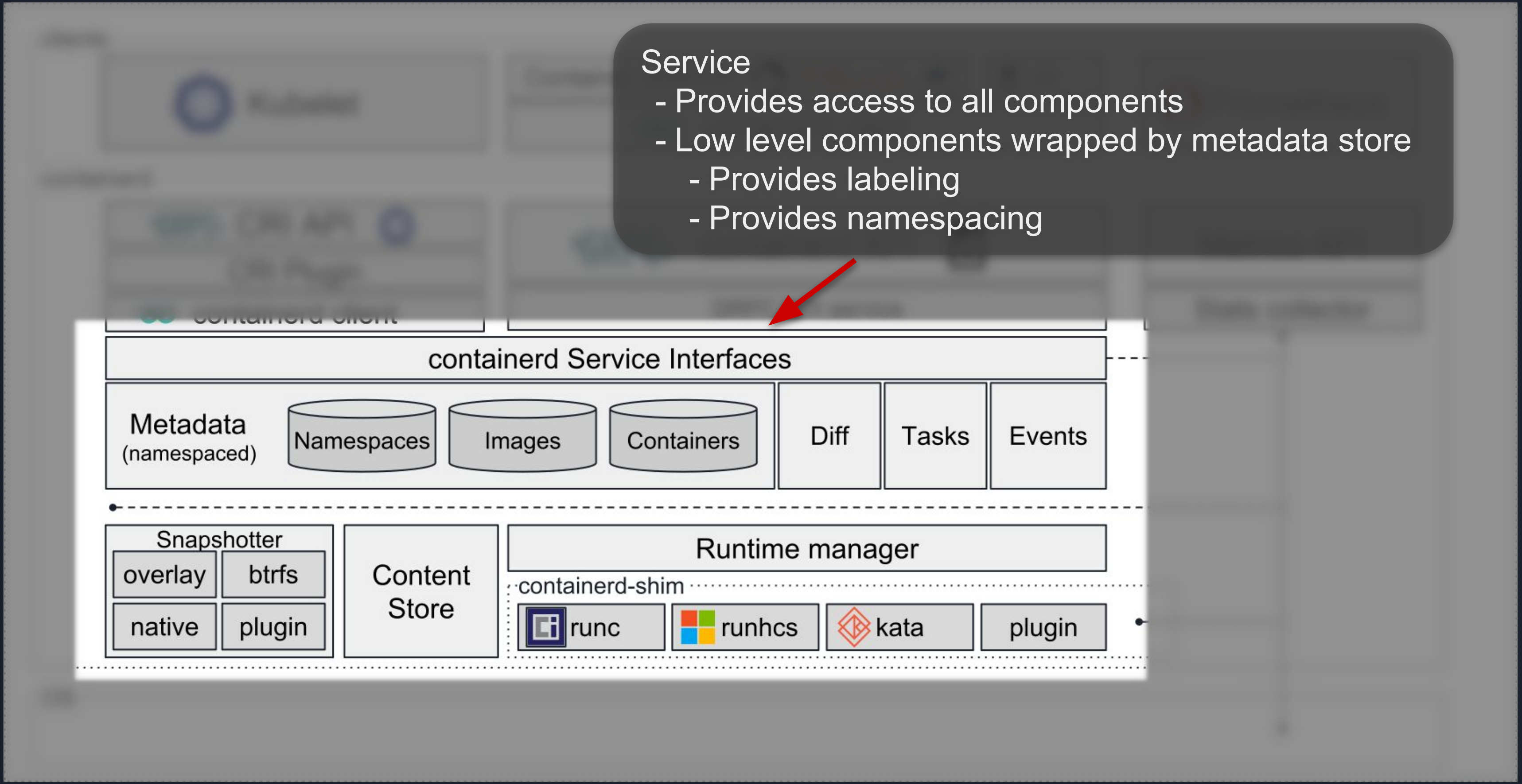
GRPC containerd API

GRPC API service

Client
- High level operations using client
- New functionality, interfaces may change (rarely)

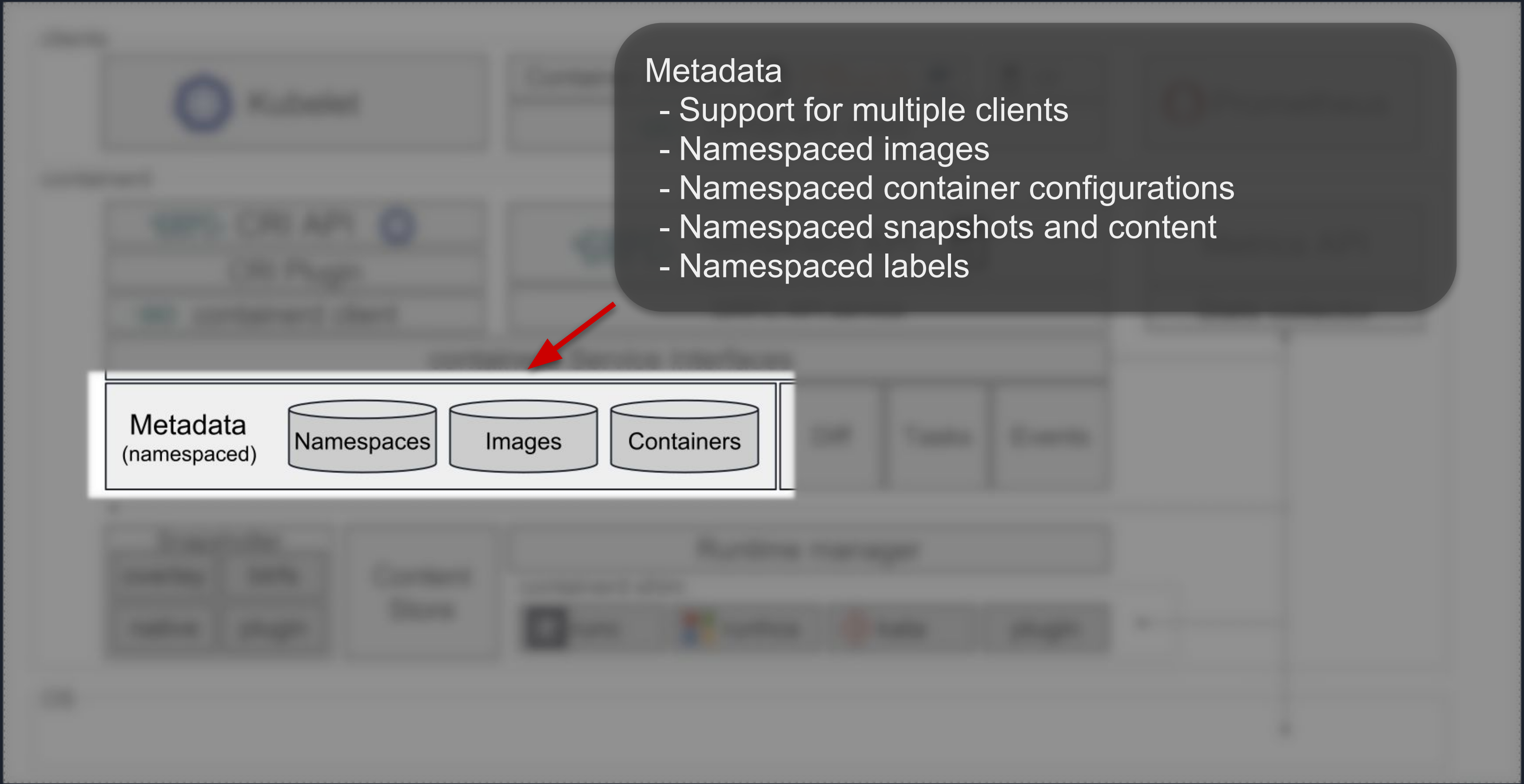
Server
- Low level interfaces to resources over GRPC
- Stable API, guaranteed 1.x compatibility

Backend Services



Service
- Provides access to all components
- Low level components wrapped by metadata store
  - Provides labeling
  - Provides namespacing

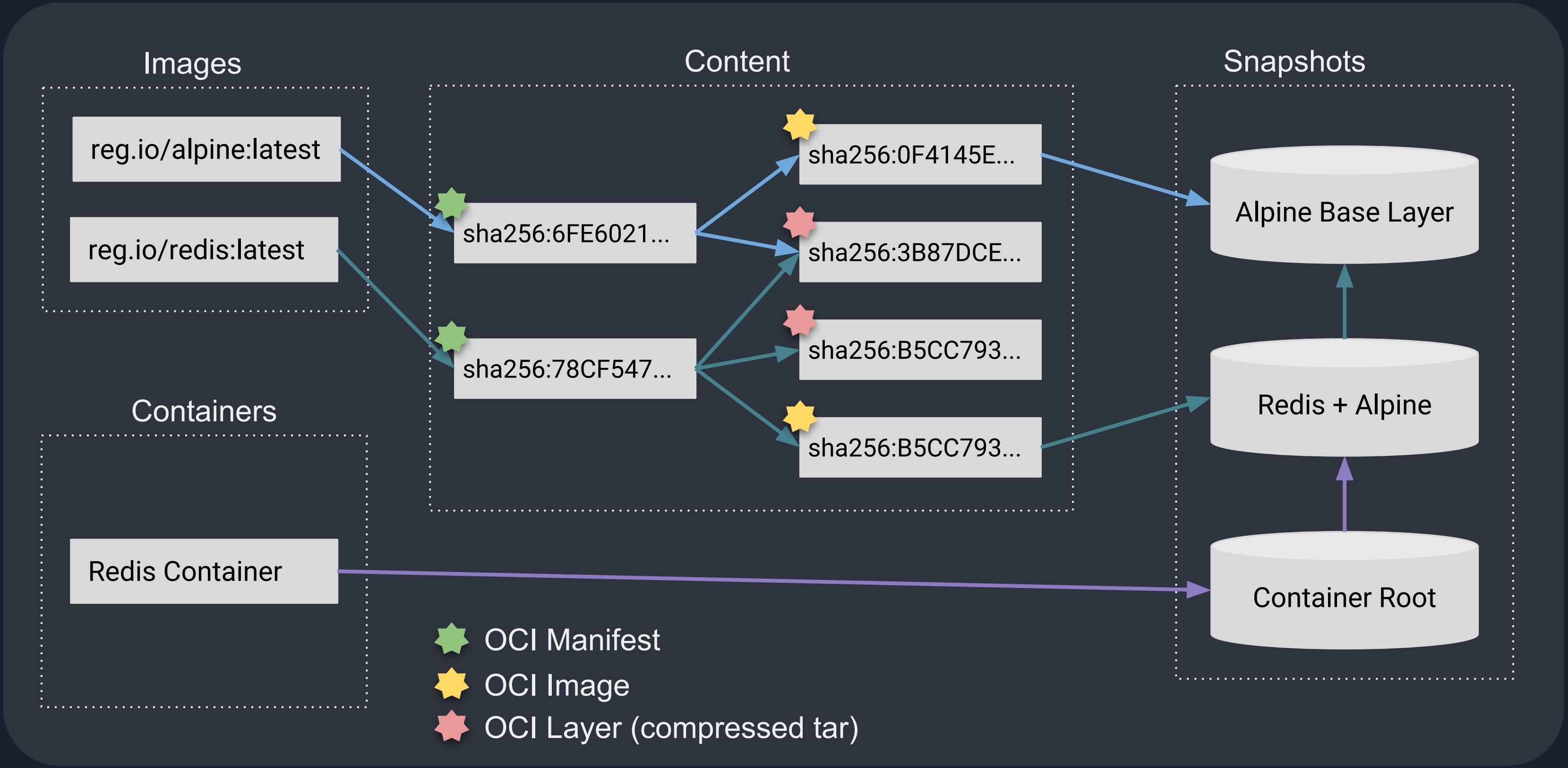
Namespacing



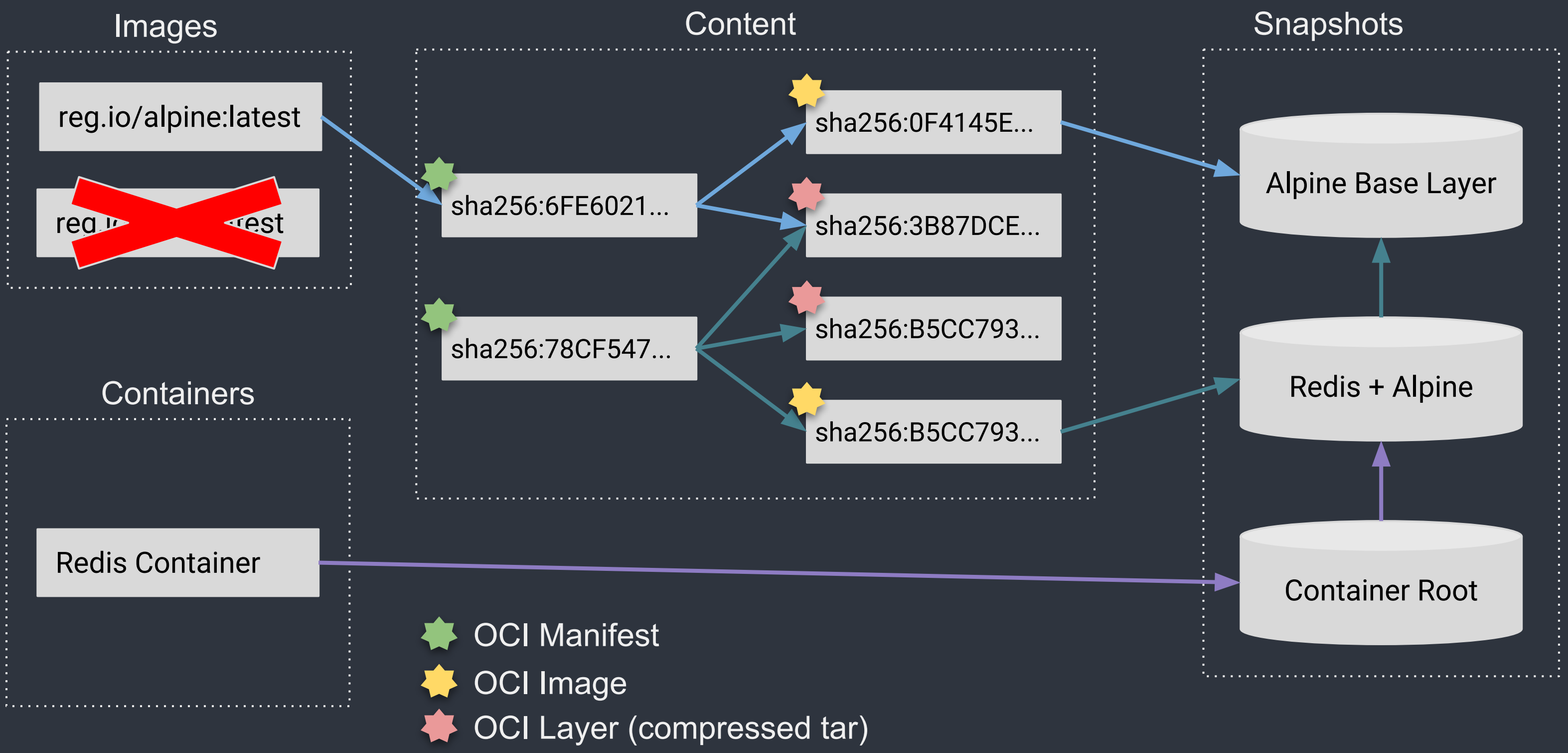
Metadata

- Support for multiple clients
- Namespaced images
- Namespaced container configurations
- Namespaced snapshots and content
- Namespaced labels

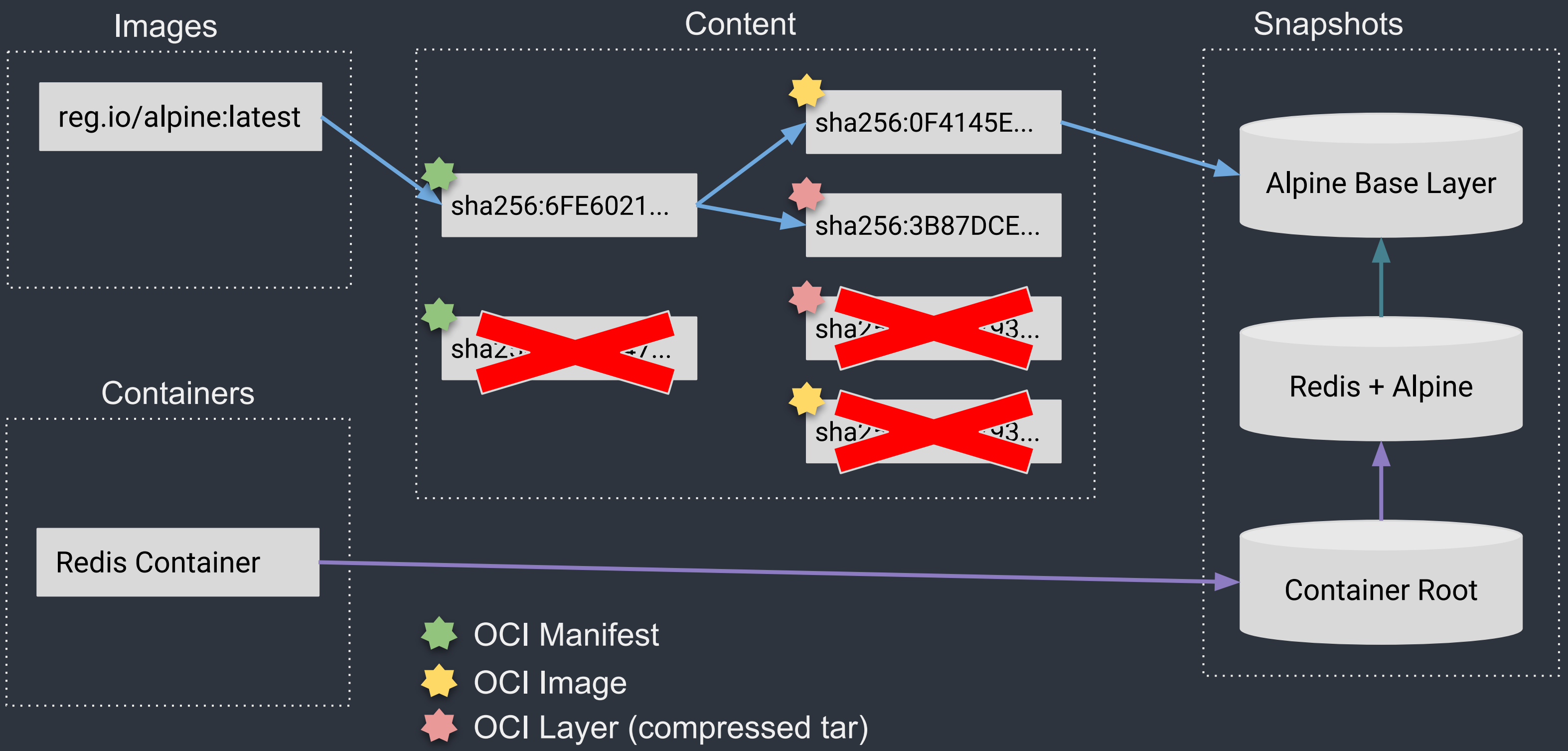
Metadata Deeper Look



Garbage Collection (delete image)



Garbage Collection (running)



Images

reg.io/alpine:latest

Content

sha256:6FE6021...

sha256:0F4145E...

sha256:3B87DCE...

~~sha2 ...7...~~

~~sha2 ...93...~~

~~sha2 ...93...~~

Snapshots

Alpine Base Layer

Redis + Alpine

Container Root

Containers

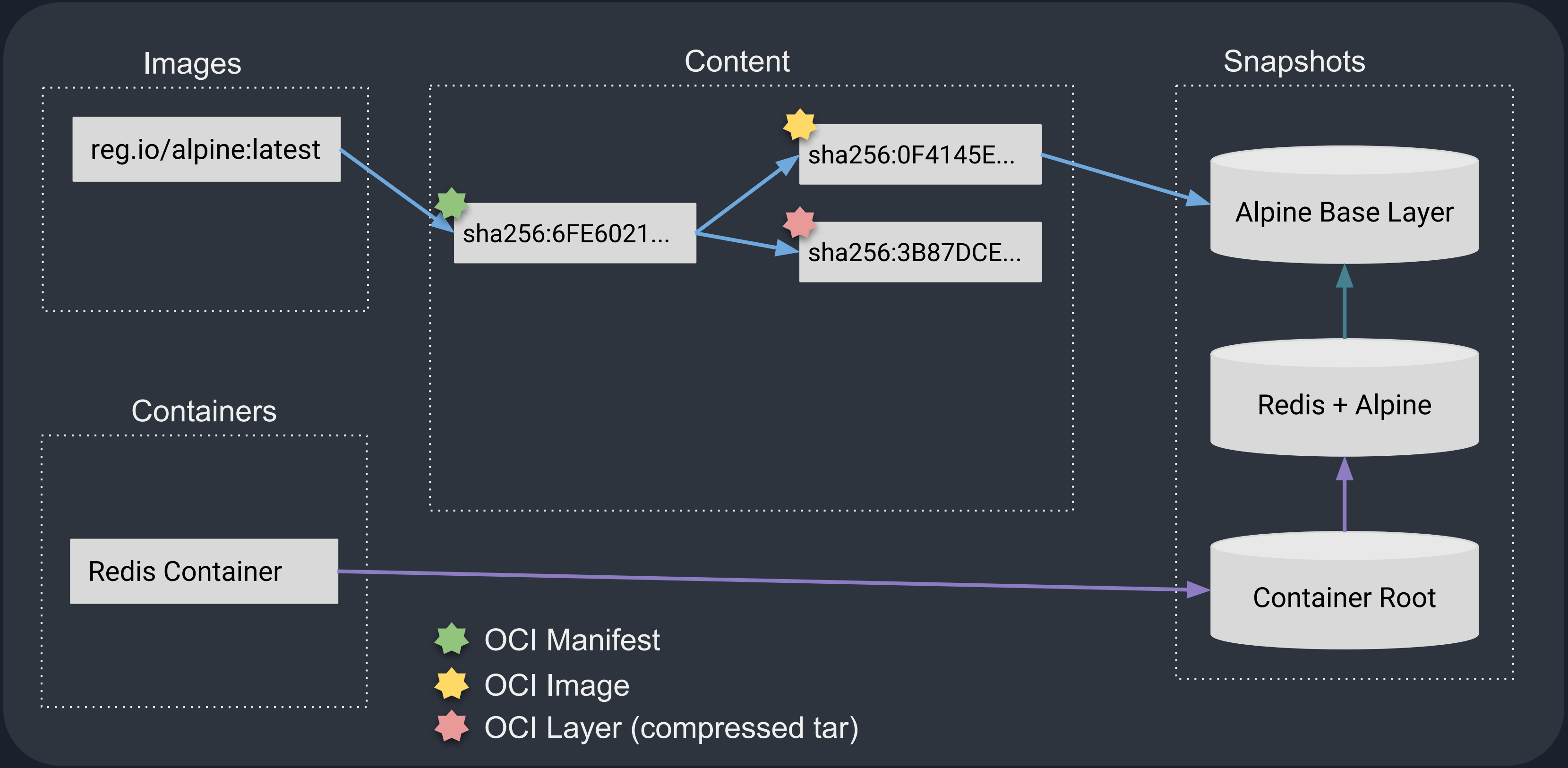
Redis Container

✱ OCI Manifest

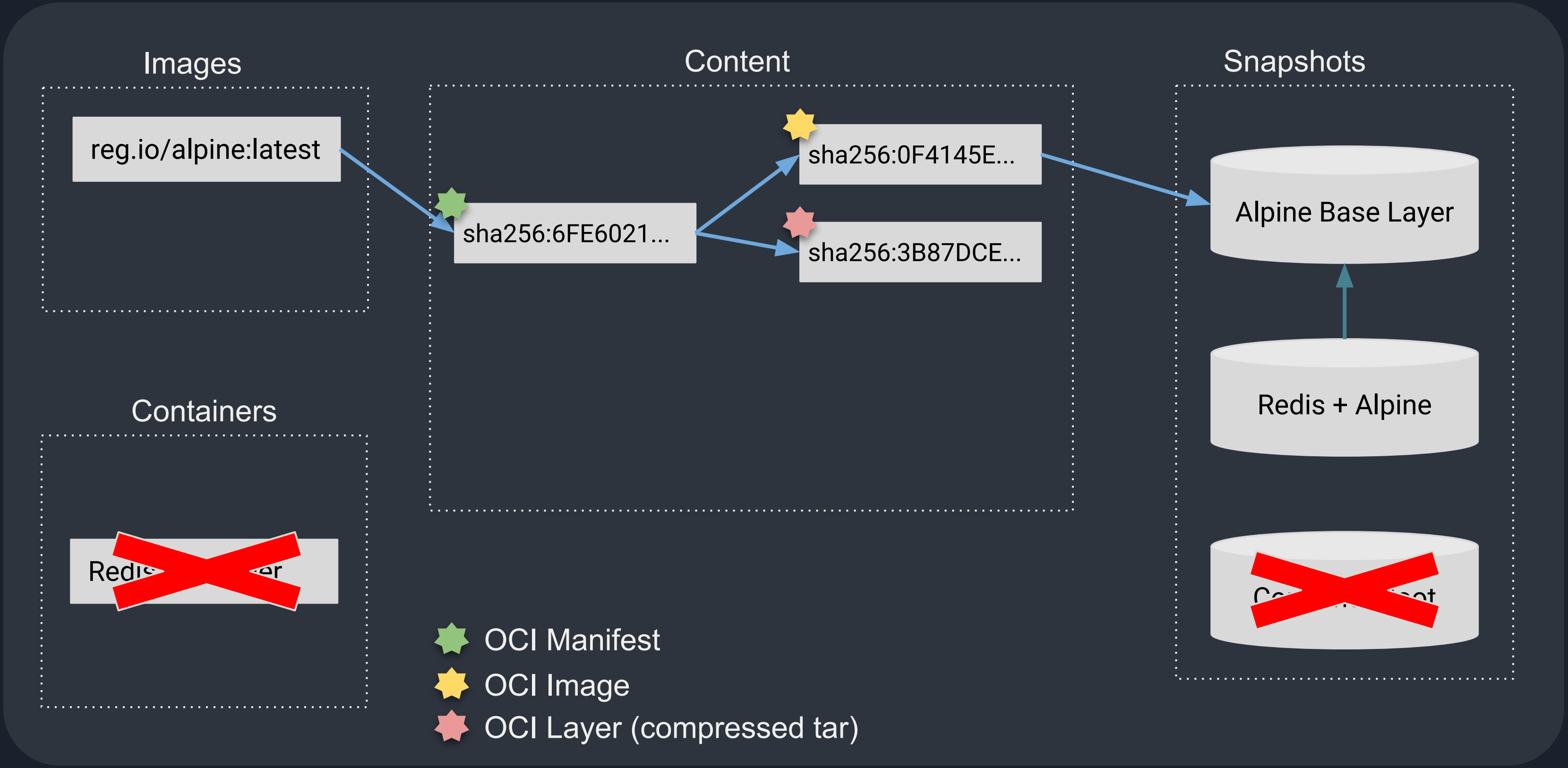
✱ OCI Image

✱ OCI Layer (compressed tar)

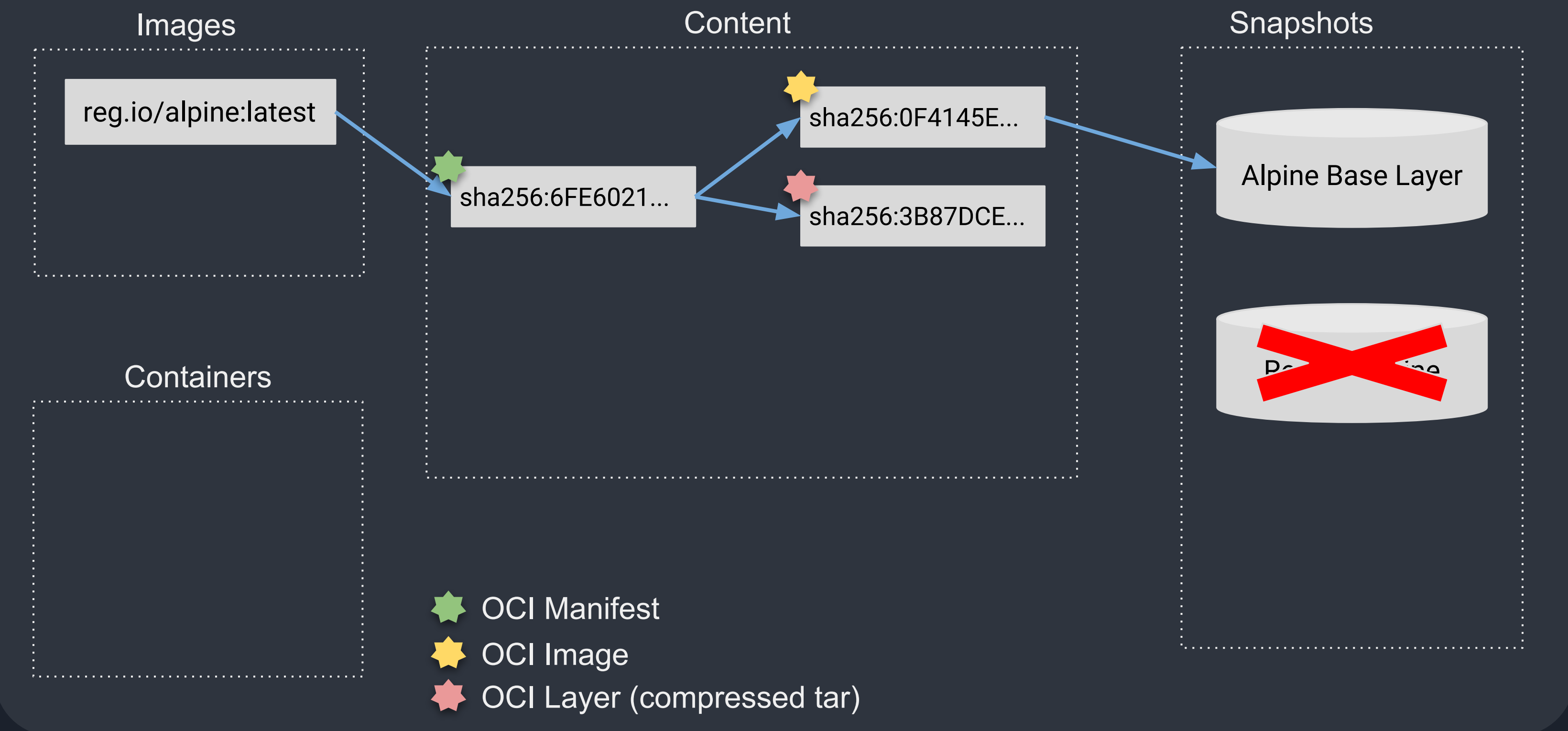
Garbage Collection (after)



Garbage Collection (delete container)



Garbage Collection (running)



Images

reg.io/alpine:latest

Containers

Content

sha256:6FE6021...

sha256:0F4145E...

sha256:3B87DCE...

Snapshots

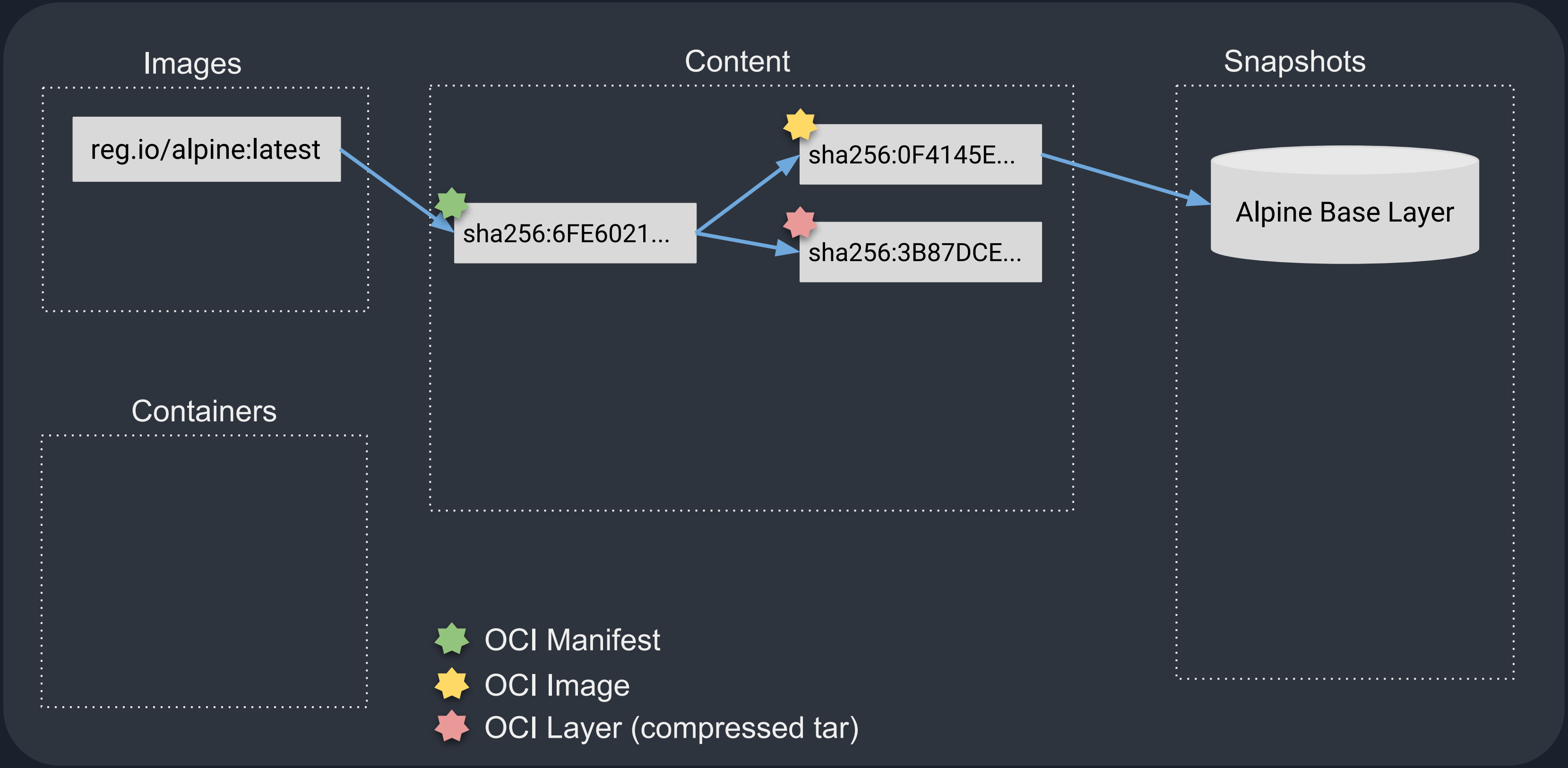
Alpine Base Layer

✸ OCI Manifest

✸ OCI Image

✸ OCI Layer (compressed tar)

Garbage Collection (final)



Images

reg.io/alpine:latest

Containers

Content

sha256:6FE6021...

sha256:0F4145E...

sha256:3B87DCE...

Snapshots

Alpine Base Layer

🟩 OCI Manifest

🟨 OCI Image

🟥 OCI Layer (compressed tar)

Plugins (snapshots)

Snapshotter Design

- No data operations
- No mounting
- Immutable snapshots
- Label support
- Enumeration

```
type Snapshotter interface {

  Stat(Context, string) (Info, error)

  Update(Context, Info, ...string) (Info, error)

  Usage(Context, string) (Usage, error)

  Mounts(Context, string) ([]mount.Mount, error)

  Prepare(Context, string, string, ...Opt) ([]mount.Mount, error)

  View(Context, string, string, ...Opt) ([]mount.Mount, error)

  Commit(Context, string, string, ...Opt) error

  Remove(Context, string) error

  Walk(Context, func(context.Context, Info) error) error

  Close() error

}
```

Plugins (snapshots)

Proxy Snapshotter

1. Build as an external plugin
2. Configure Containerd to use proxy plugin (/etc/containerd/config.toml)

1

```
package main

import(
  "net"
  "log"
  "github.com/containerd/containerd/api/services/snapshots/v1"
  "github.com/containerd/containerd/contrib/snapshotservice"
)

func main() {
  rpc := grpc.NewServer()
  sn := CustomSnapshotter()
  service := snapshotservice.FromSnapshotter(sn)
  snapshots.RegisterSnapshotsServer(rpc, service)

  // Listen and serve
  l, err := net.Listen("unix", "/tmp/sn.sock")
  if err != nil {
    log.Fatalf("error: %v\n", err)
  }
  if err := rpc.Serve(l); err != nil {
    log.Fatalf("error: %v\n", err)
  }
}
```

2

```
...
[proxy_plugins]
  [proxy_plugins.mysnapshotter]
    type = "snapshot"
    address = "/tmp/sn.sock"
...
```

Plugins (runtime)

Shim GRPC API

- Low level
- Stats now included
- Stabilized in 1.2

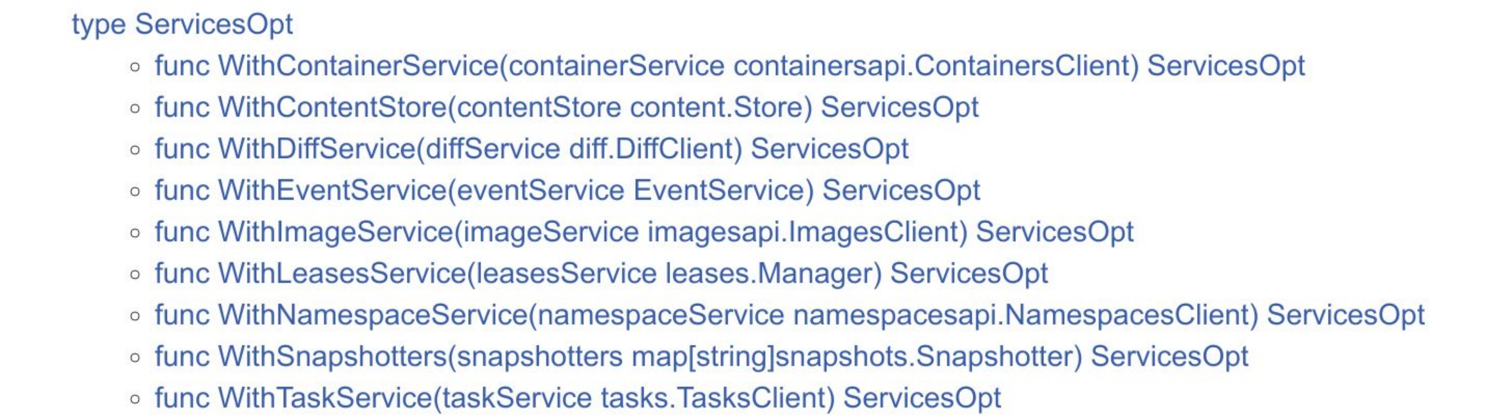
Shim Processes

- At most 1 per container
- Can be shared (id passed to every request)

```
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);
}
```

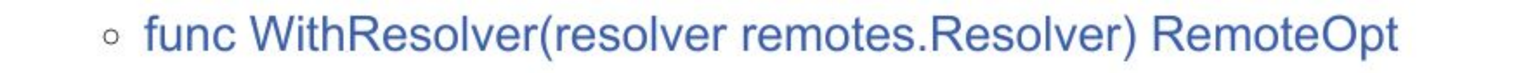
Client Extensibility

Use your own services with service options



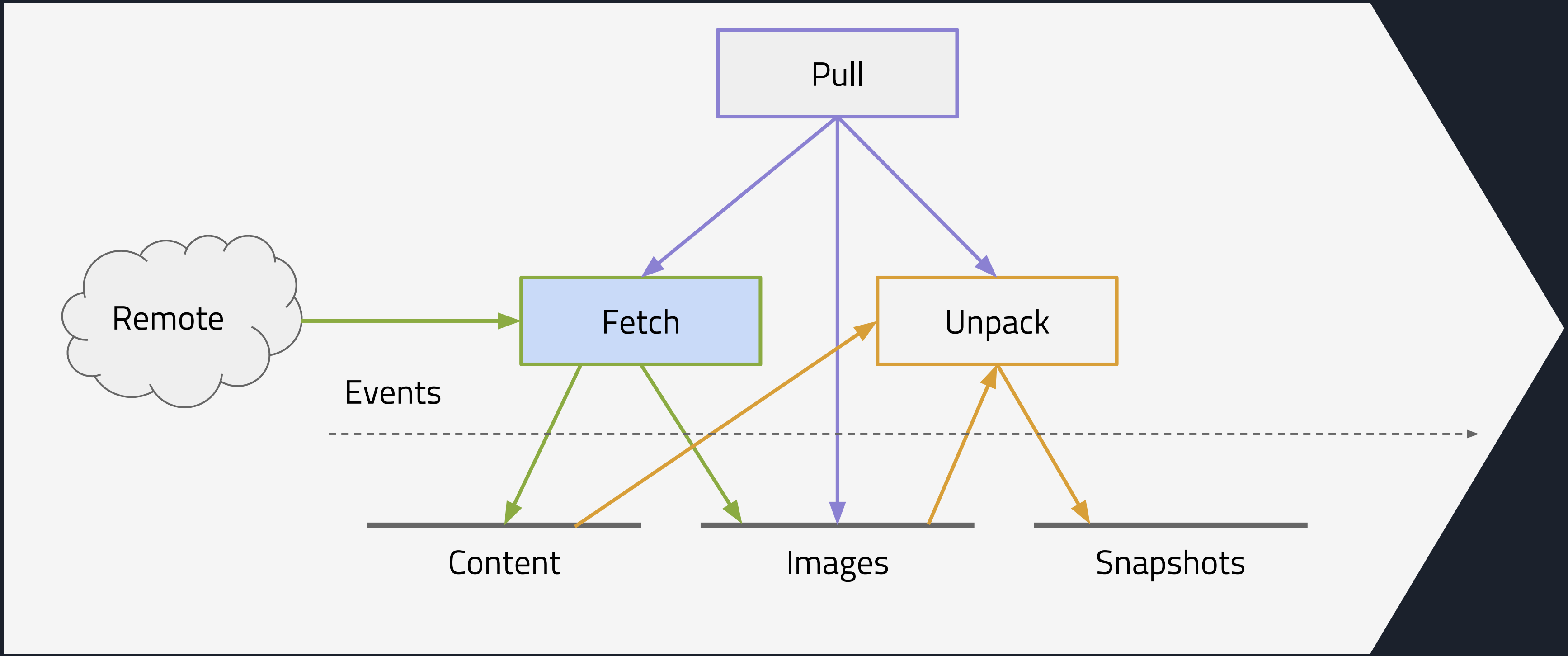
- type ServicesOpt
  - func WithContainerService(containerService containersapi.ContainersClient) ServicesOpt
  - func WithContentStore(contentStore content.Store) ServicesOpt
  - func WithDiffService(diffService diff.DiffClient) ServicesOpt
  - func WithEventService(eventService EventService) ServicesOpt
  - func WithImageService(imageService imagesapi.ImagesClient) ServicesOpt
  - func WithLeasesService(leasesService leases.Manager) ServicesOpt
  - func WithNamespaceService(namespaceService namespacesapi.NamespacesClient) ServicesOpt
  - func WithSnapshotters(snapshotters map[string]snapshots.Snapshotter) ServicesOpt
  - func WithTaskService(taskService tasks.TasksClient) ServicesOpt

Customize push and pull with remote options

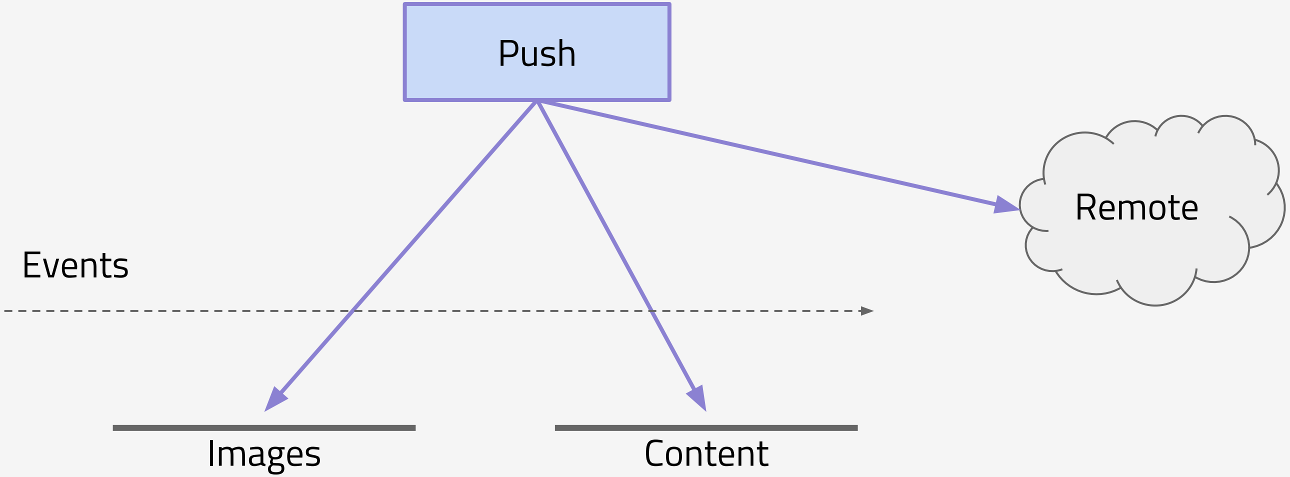


- func WithResolver(resolver remotes.Resolver) RemoteOpt

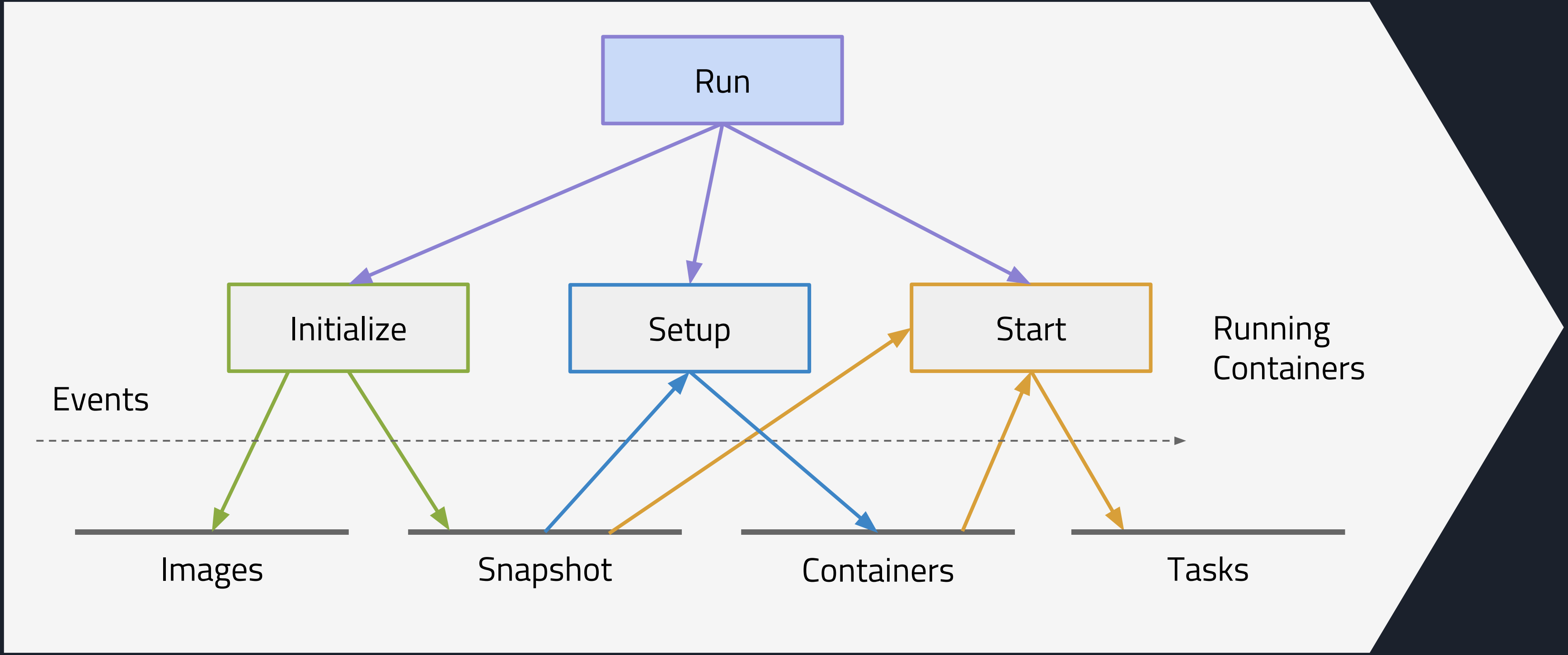
Flows (pull)



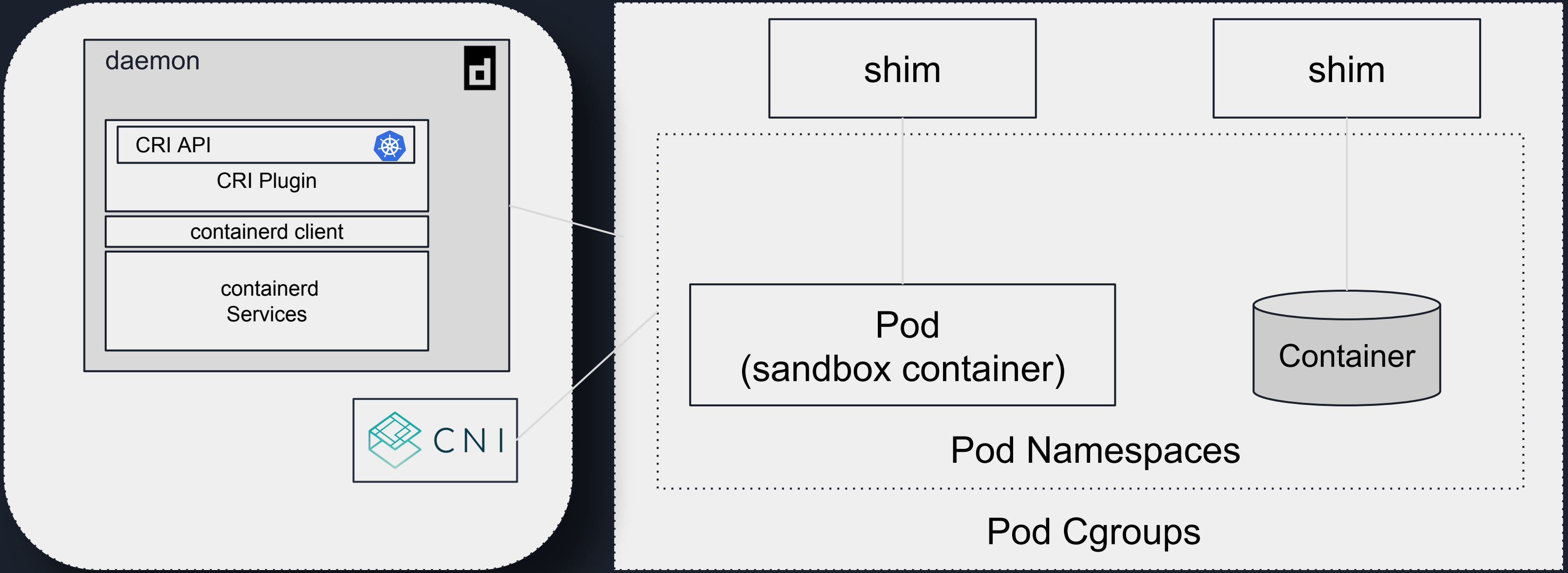
Flow (push)



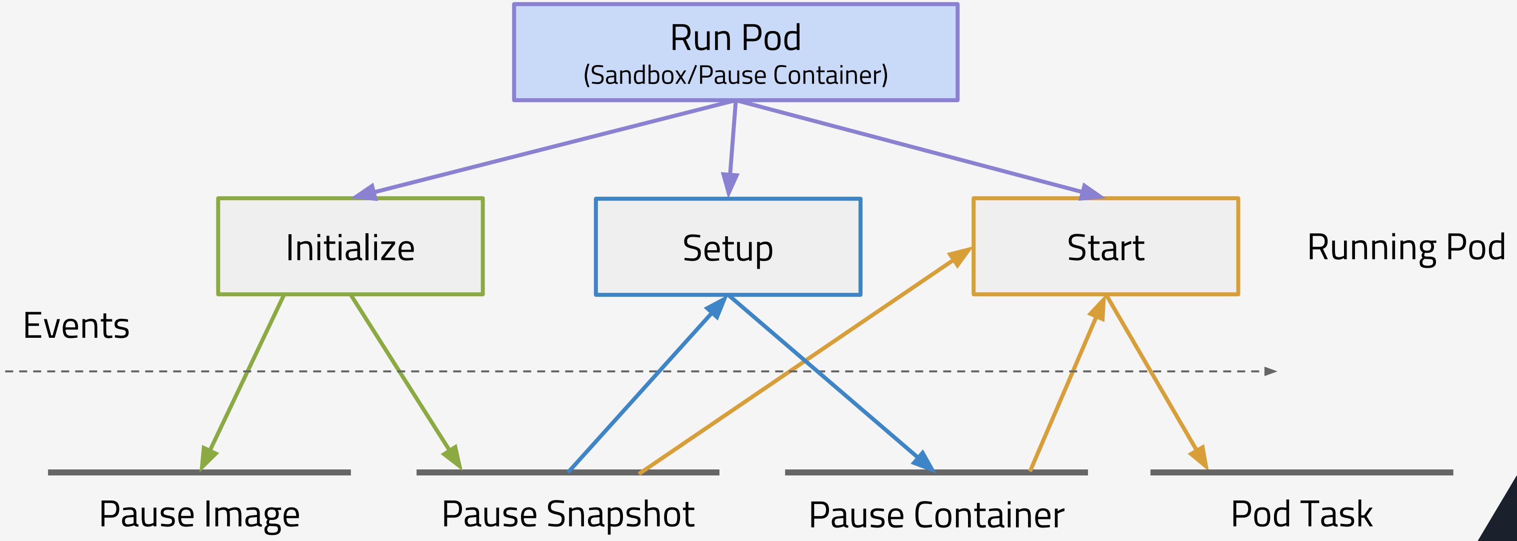
Flow (container run)



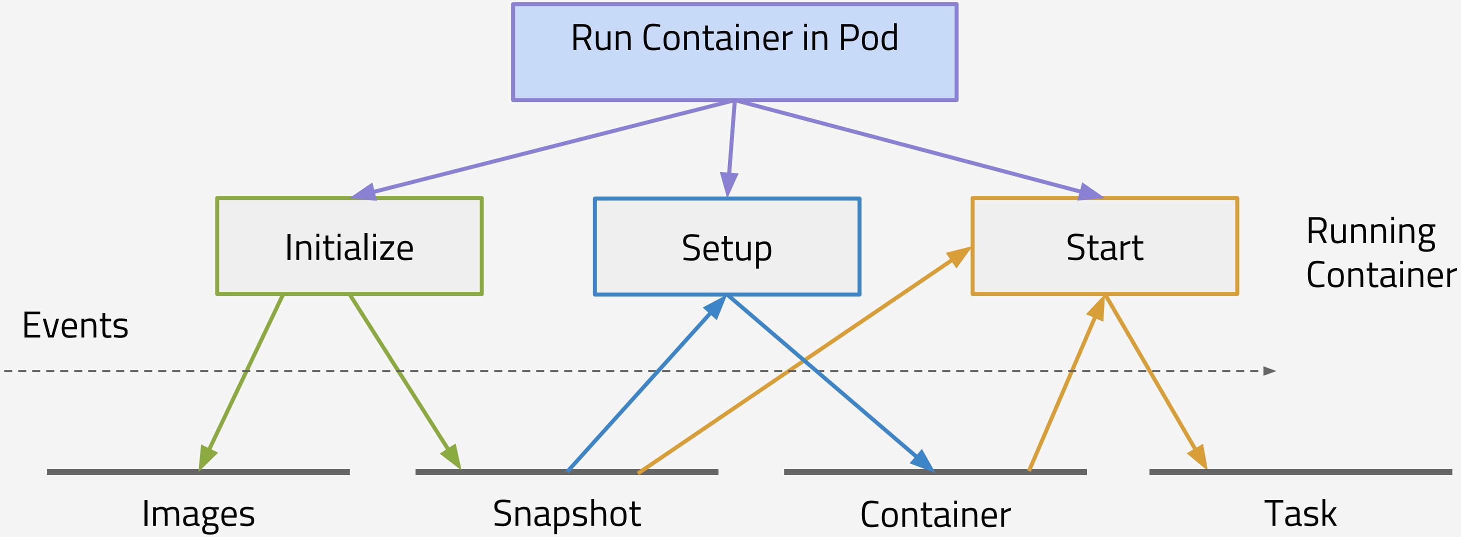
CRI - Pod Namespaces



CRI pod Flow (run pod)



CRI pod Flow (run container)



Demo