**Monitoring Solution**

**High level:**

The application stack implementation of Alerts and Prometheus managed services will be based on the operator pattern. The operators will be able to work on both K8s or Cattle. These operators will provide the following core functionality:

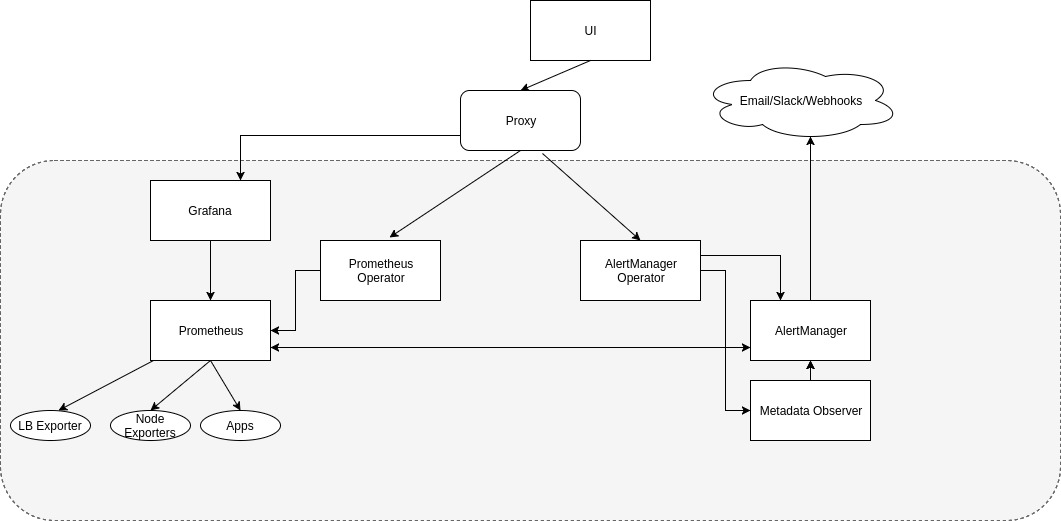
1. Present an API that will be mapped via the magic proxy /r/ paths in Rancher.

2. Store data either in Cattle DB and/or K8s back ends as key value pairs.

a. Cattle generic objects need to be updated to support this.

3. If monitoring is enabled, Rancher load balancers will need to be deployed with stats on, this will either need to be done via custom HAProxy configurations or compose. The stats will need to expose an endpoint reachable by the haproxy\_exporter for Prometheus.

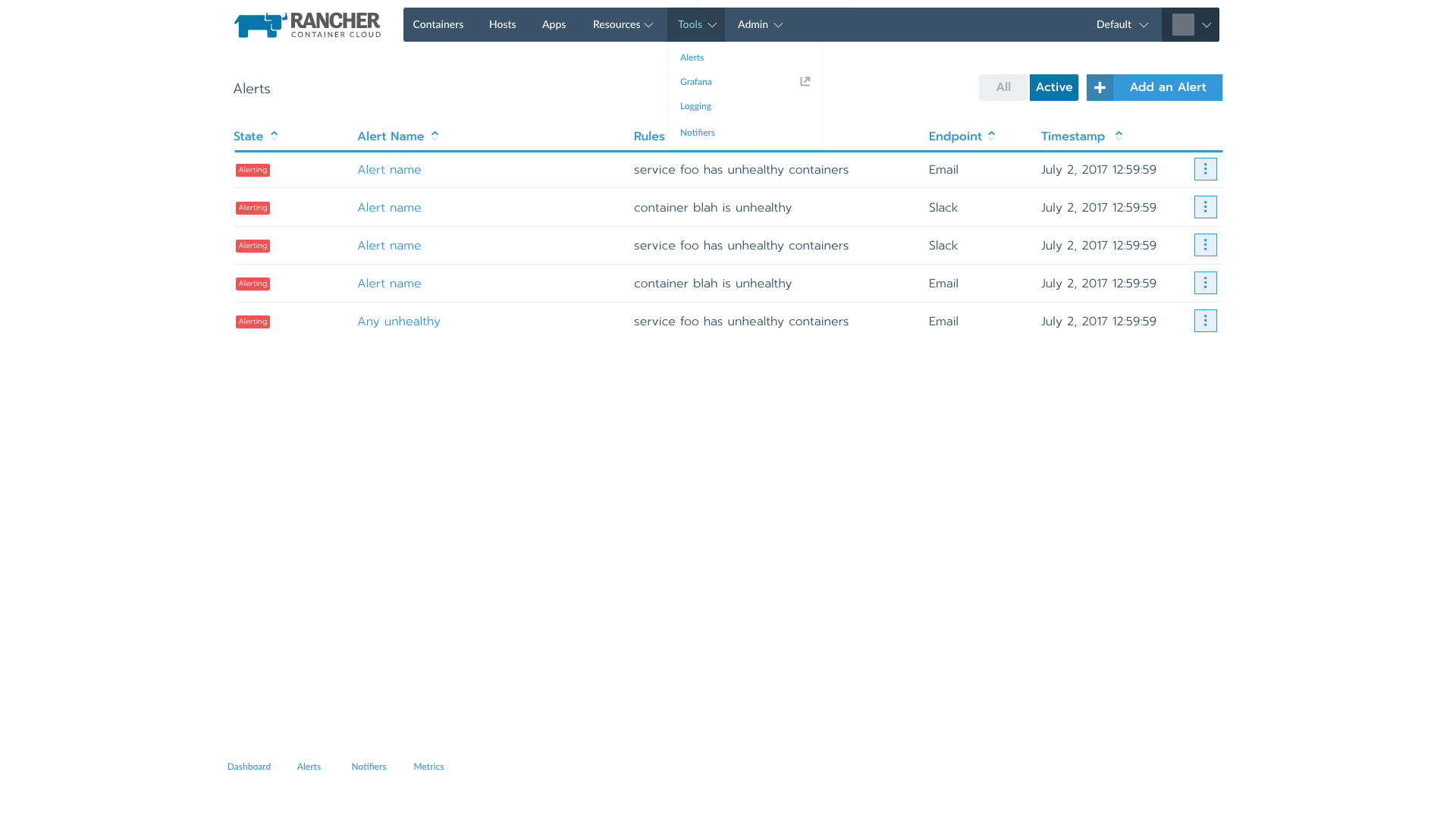
**Architecture:**



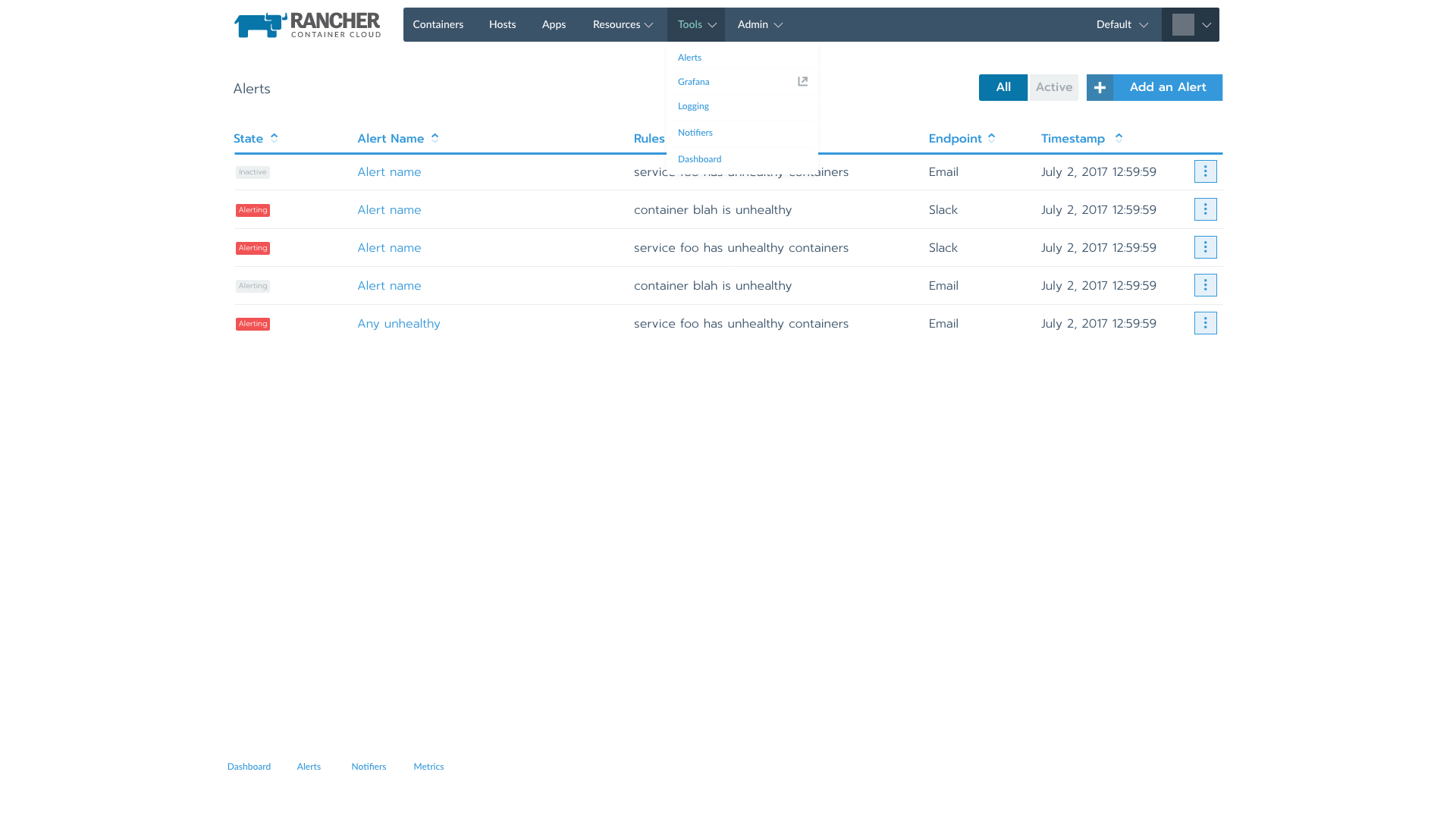
The UI will add a new top level menu “Tools”, or something similar, to house new additions such as notifications, alerts, monitoring and dashboard. This heading is more specific to the area of DevOps tooling and does not fit into the “Resource” category. From this section, the user will be able to visualize monitoring data and configure various settings.

**Alerting**

Users will be able to see alerts out of the box from Rancher based services in a new page called “Alerts”. There is an “Active” filtered view of “All” alerts that will be set as the default when events are alerting. The active state will show alerts that are actively experiencing an event. Without a notifier configured, this will be the only place the events are seen. This data will be provided by the Prometheus AlertManager, or via proxy from the Operator.

*Active alert events.*

All states of alerts will be visible and configurable via the “All” view on the page. The configuration of all alerts will be stored in the Rancher database. The configuration will be consumed by the operator to configure the AlertManager appropriately.



*All alerts*

Initial alerts will be configured based on health checks within Rancher out of the box. The health checks built into Rancher already provide a robust monitoring solution within the environment. The checks are run from 3 separate hosts within Rancher environments. Rancher will support multiple health checks and actions per container service.

Container rules

1. Single container unhealthy.

Service Rules

1. Any percentage of the containers are unhealthy.

Host Rules

1. Any percentage of hosts are in a disconnected state.

2. If X host went goes down.

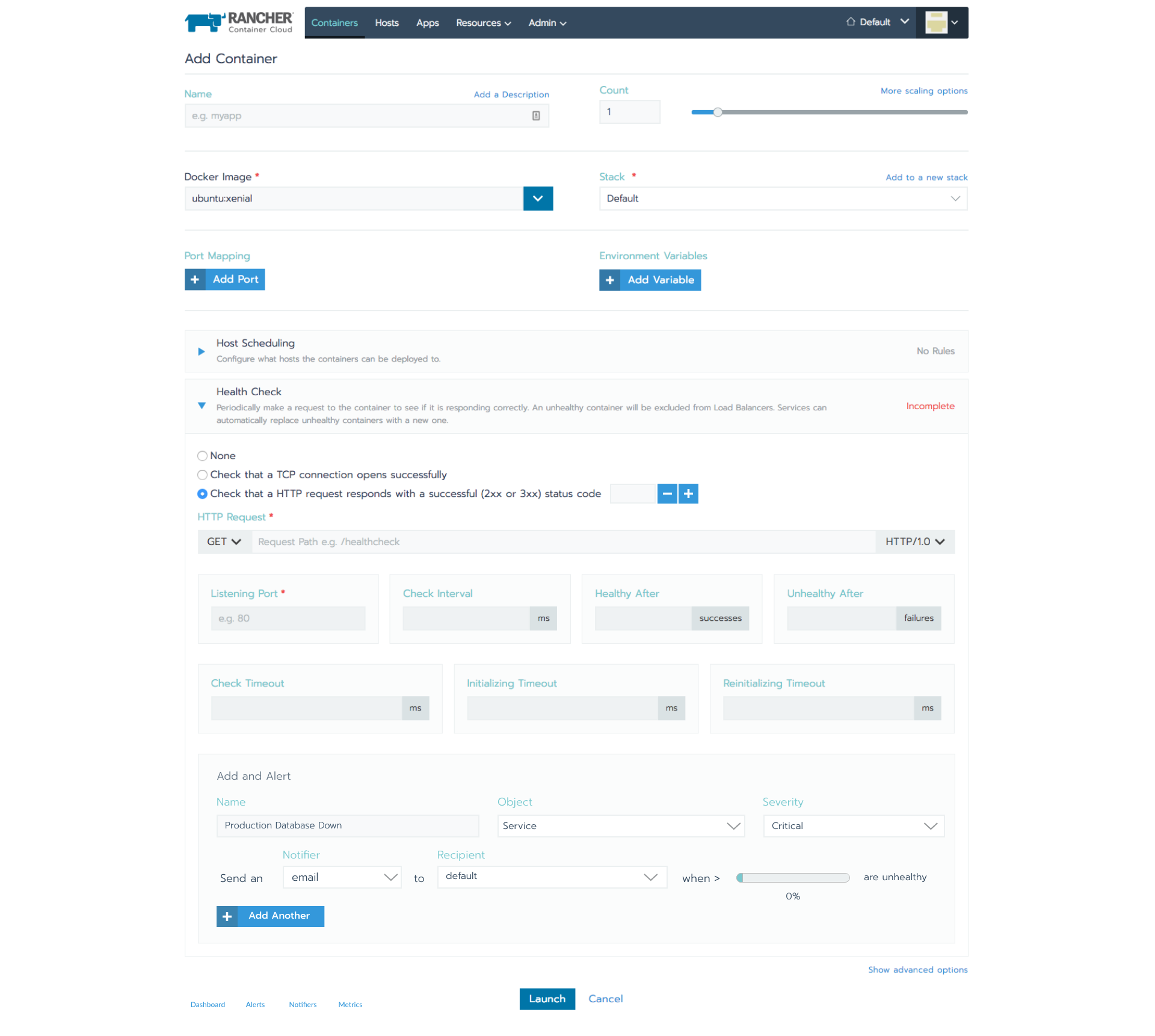
3. If host with Y label is some % gone.

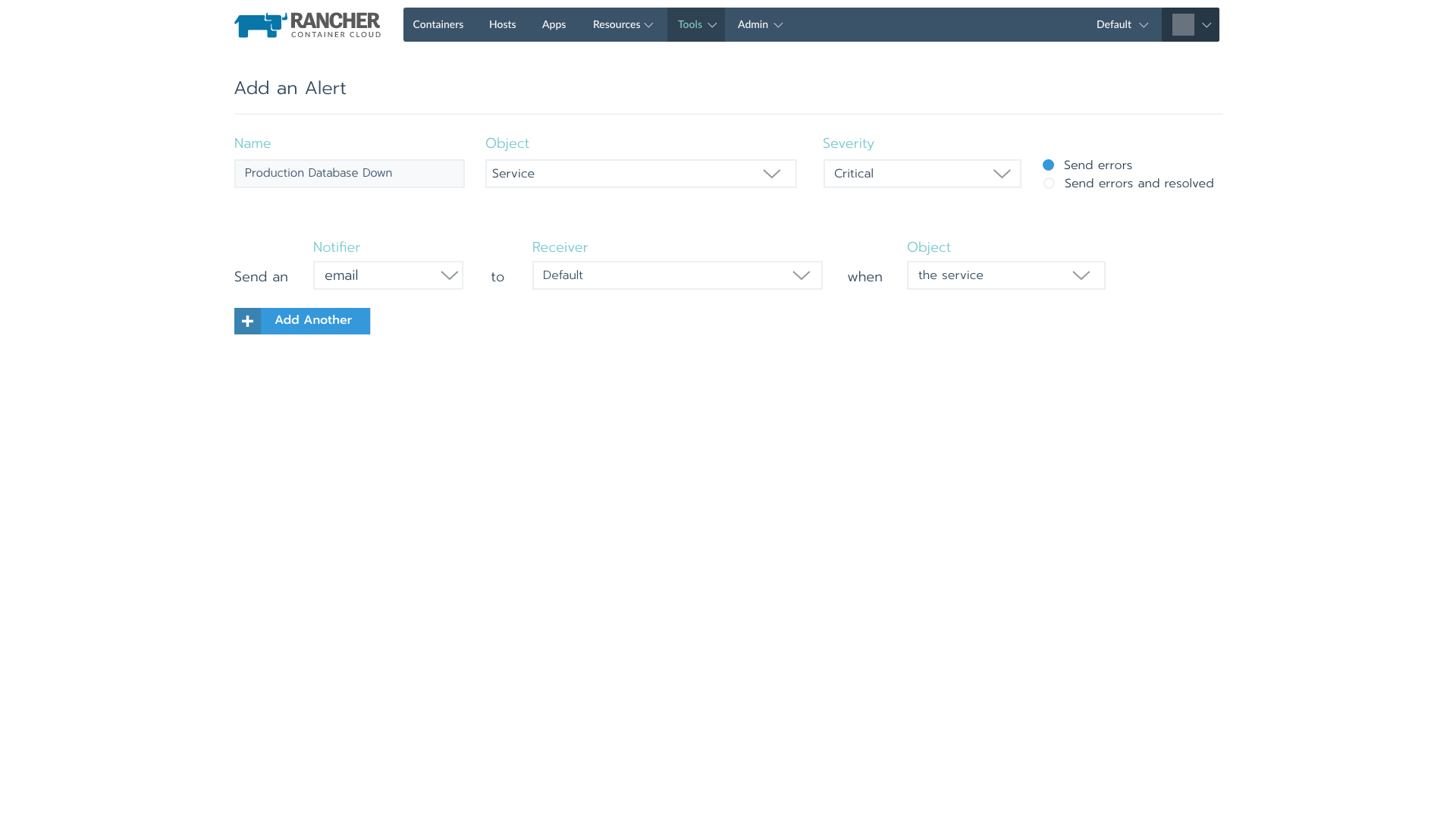
4. Disk capacity monitoring.

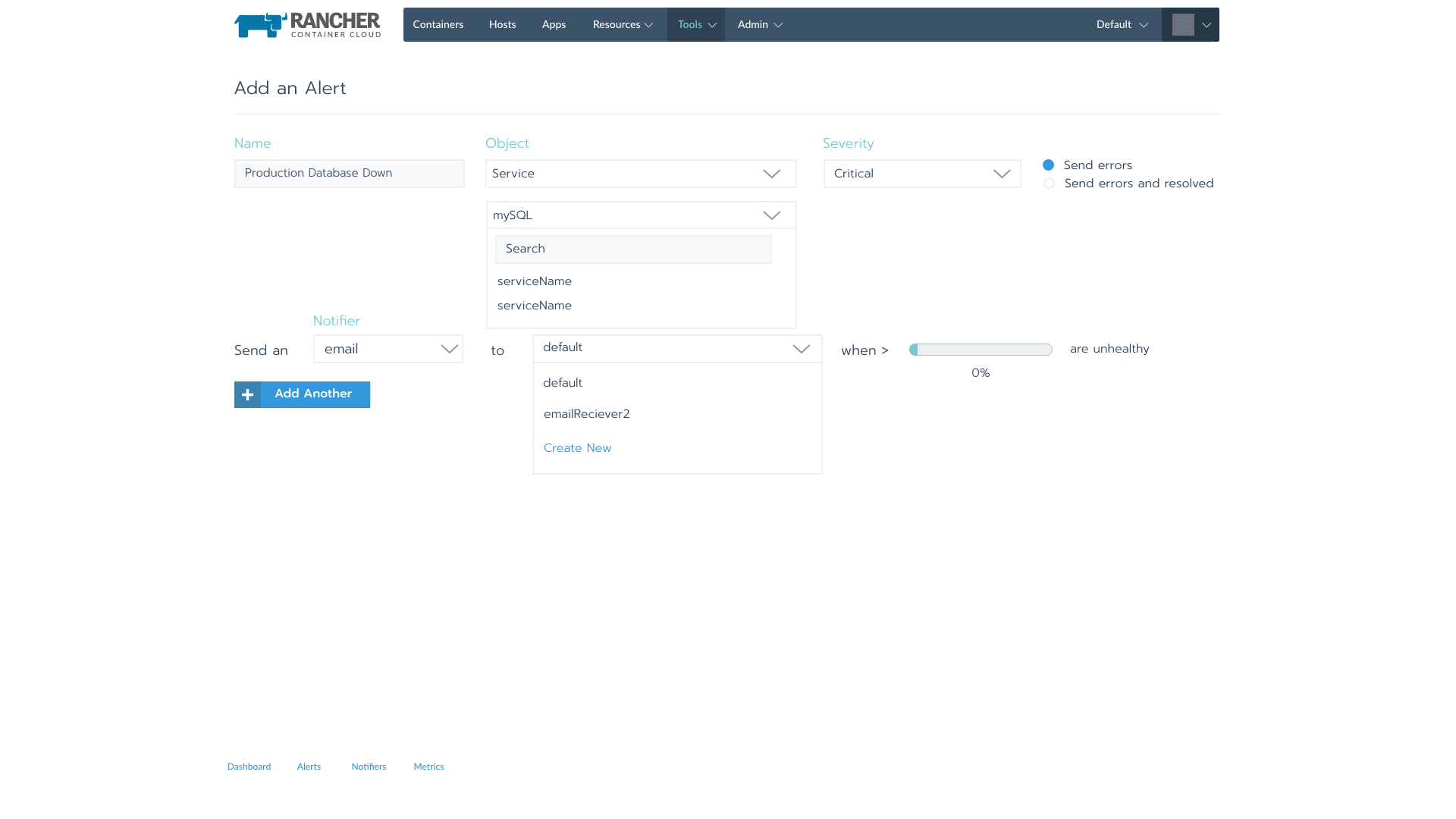
Stacks

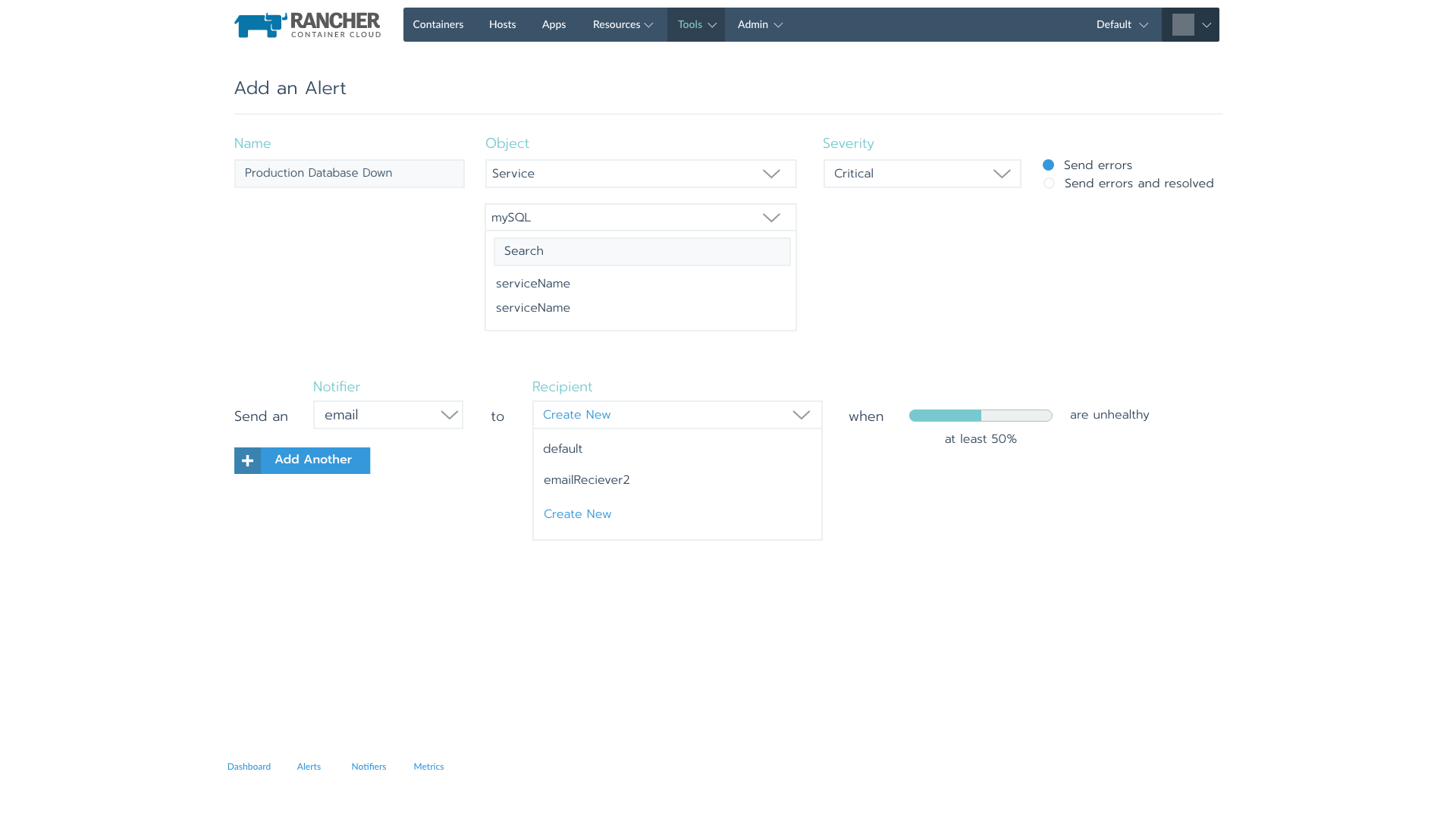
1. Any percentage of services are unhealthy.

Service rules will be configured both in the health check section when launching the service/container, and also from the add alert page.

*Add alert from container/service health check*

*Add alert from “Alert” page*

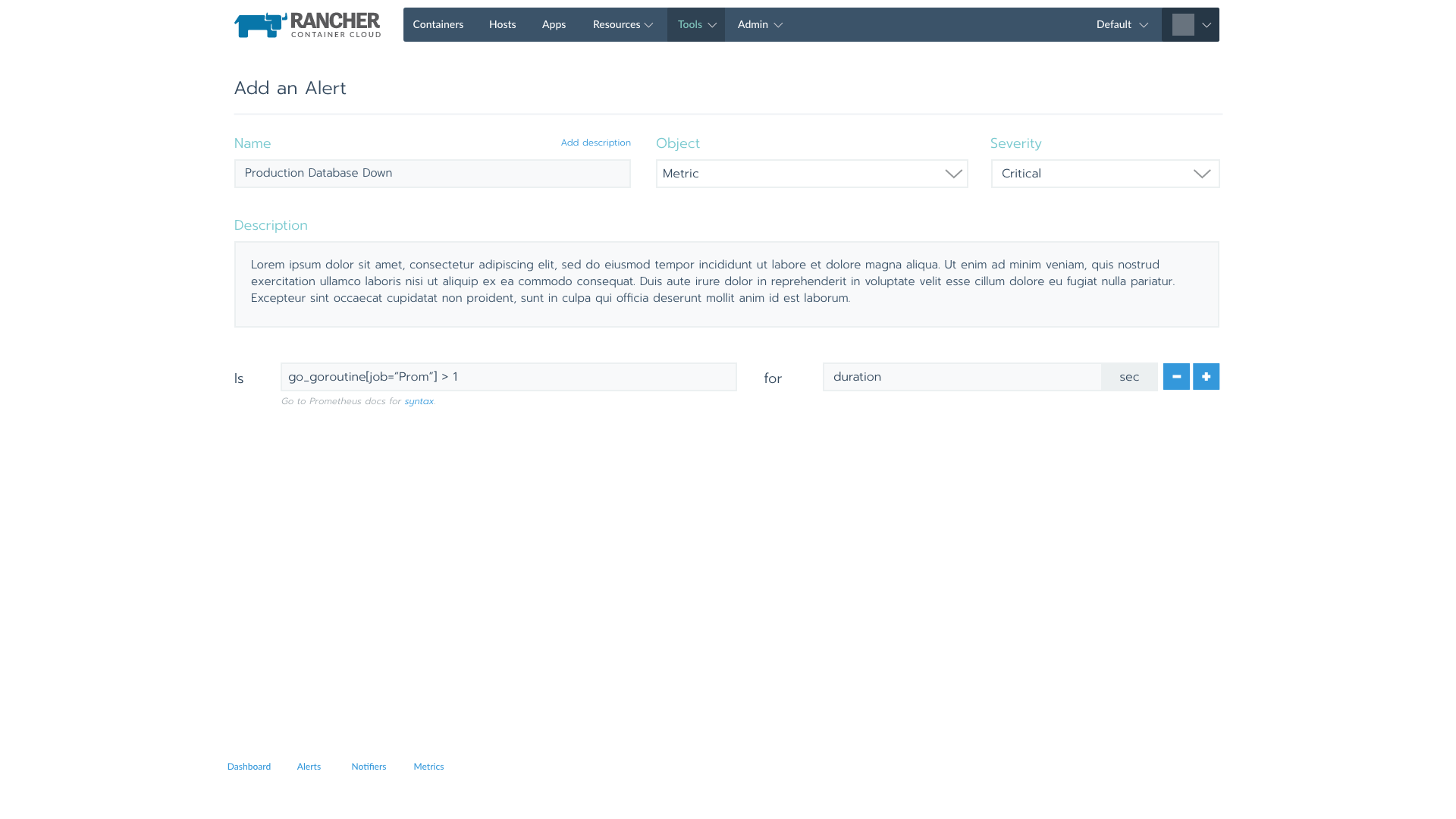
*Add alert from “Alert” page cont.*



*Add alert from “Alert” page cont.*

**Alerting with Monitoring Solution enabled:**

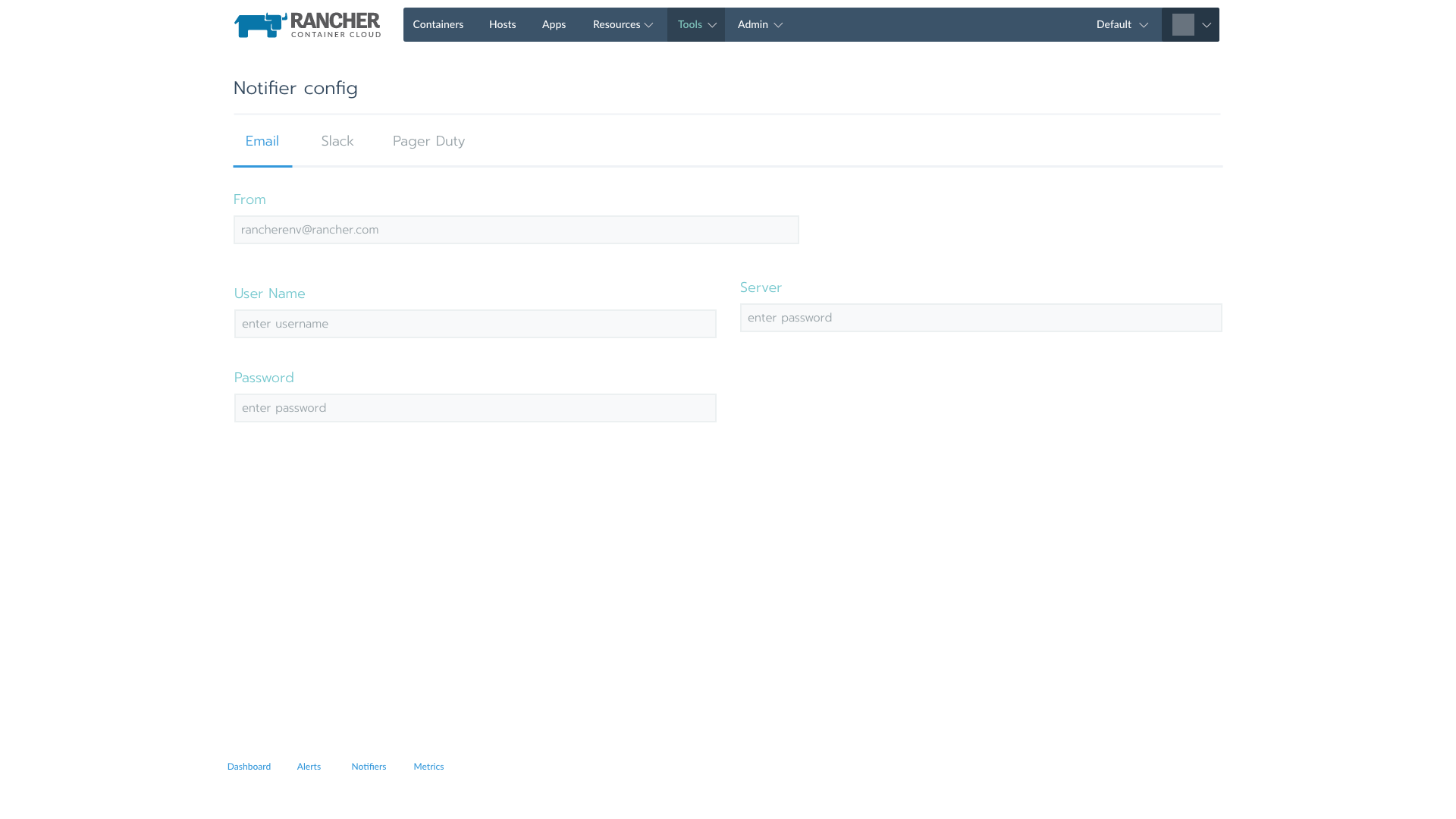
When the monitoring solution is deployed, via managed Prometheus catalog entry, the user will be able to configure custom monitoring endpoints. Based on those queries the user will be able to set alerts based on application level telemetry.

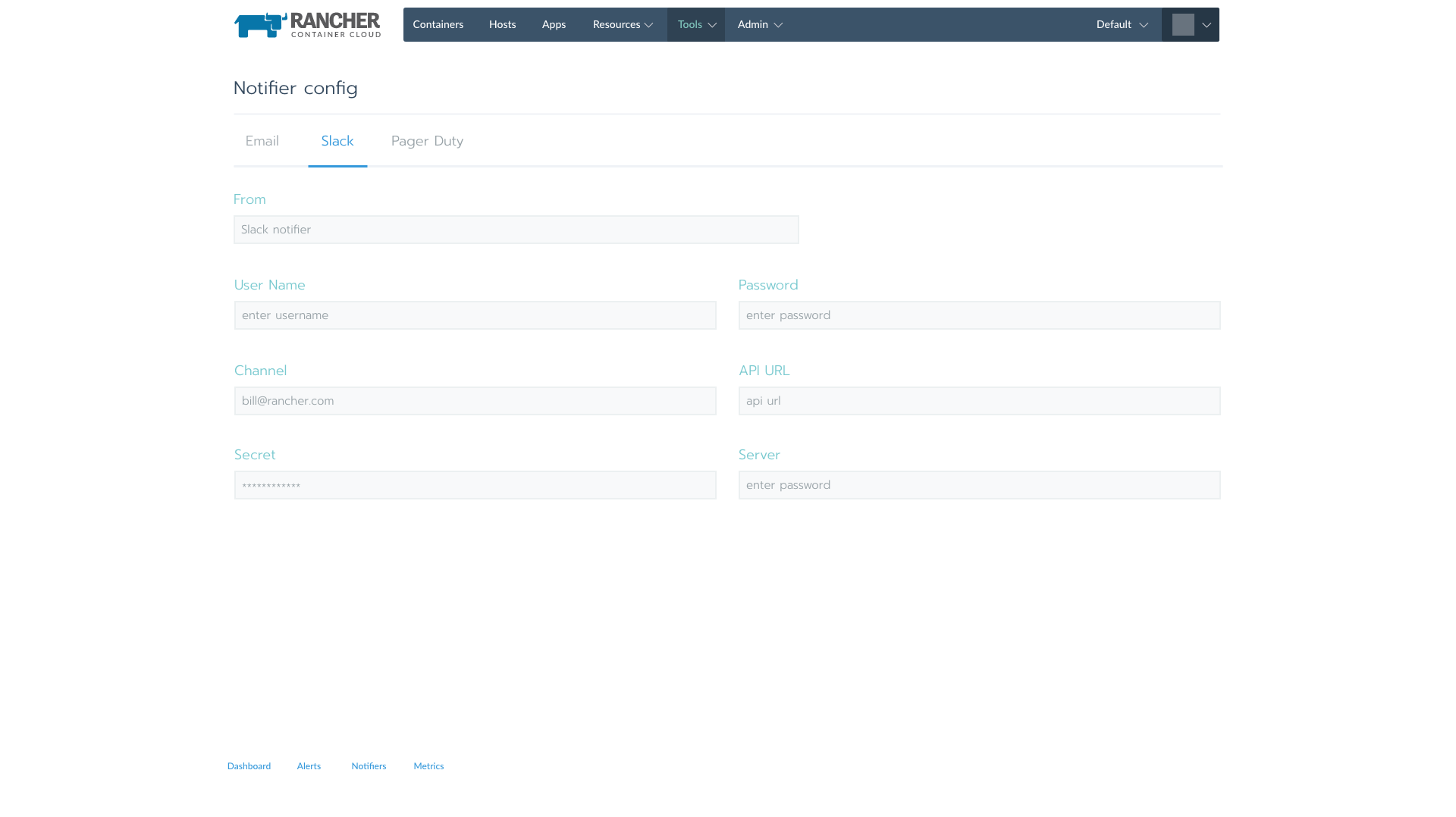
*Add alert based on custom metrics*

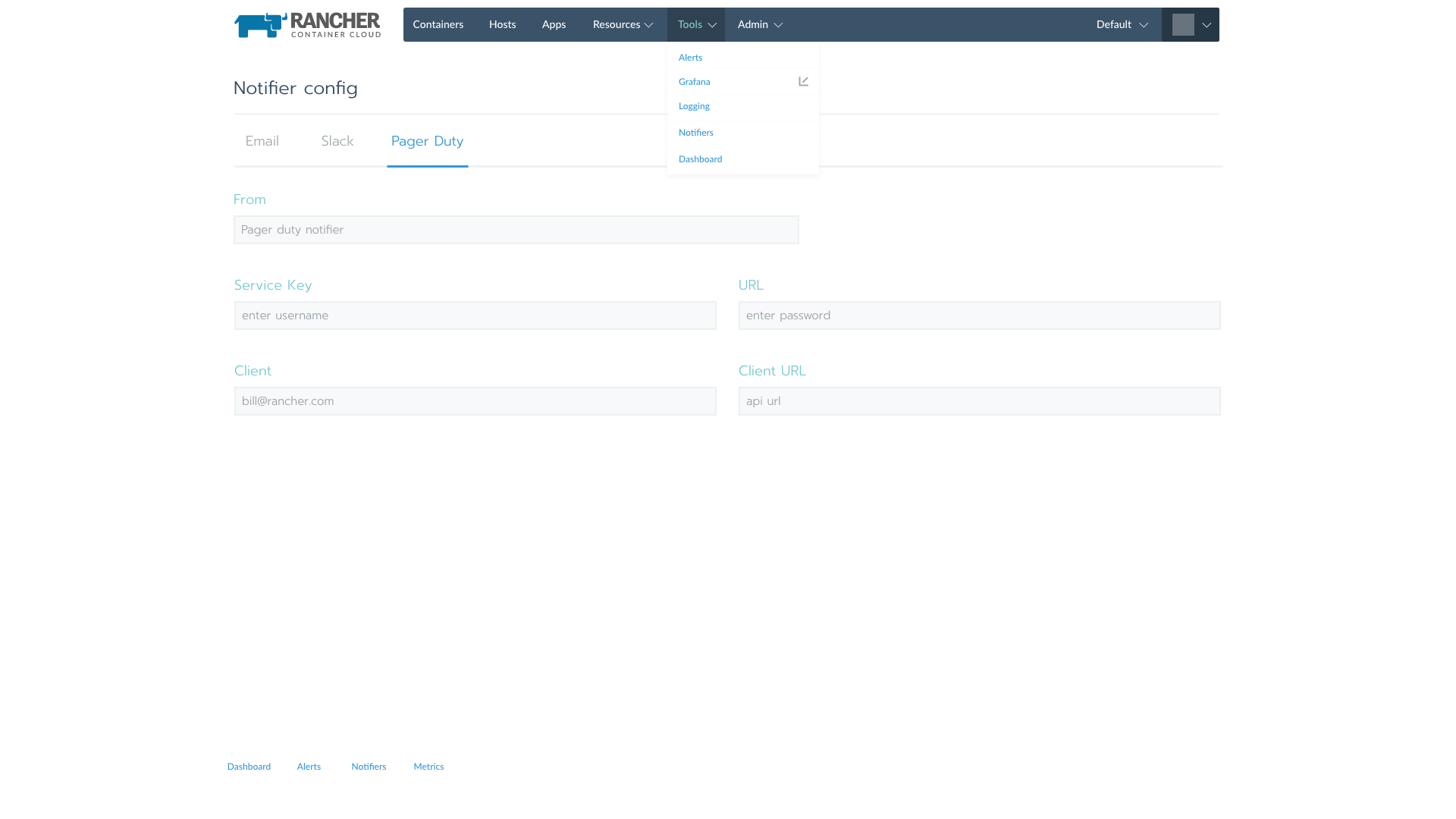
Out of the box the user will be able to immediately use the Rancher load balancing statistics to Alert based on response codes, byte sizes and response times.

**Notifications:**

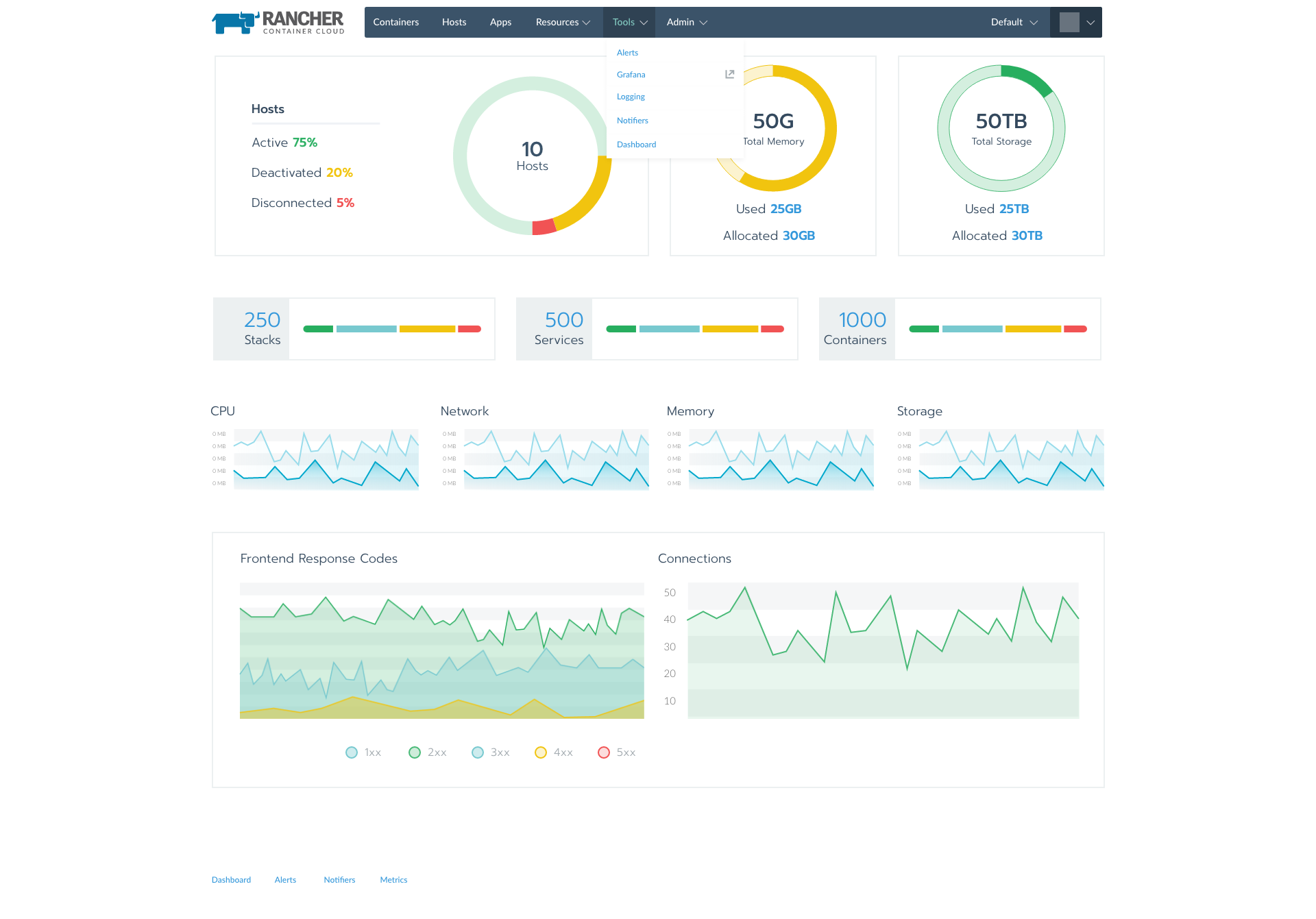
A user will be able to setup alerting without installing any additional monitoring solutions. The user will configure the notifier settings in a new settings page per environment under the Operations tab. The default endpoints will include email, Slack, webhooks and PagerDuty to start.



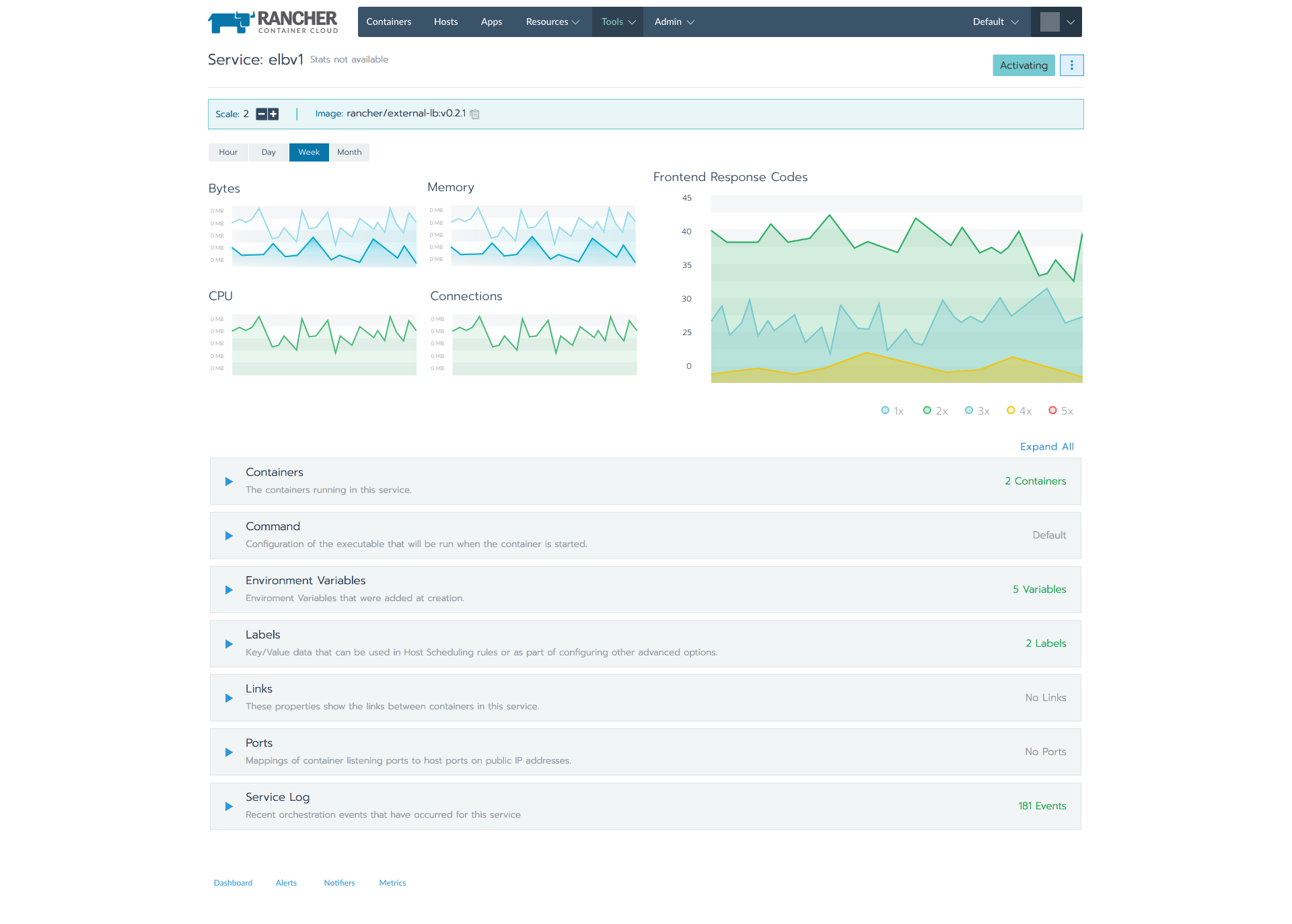


**Monitoring and Stats**

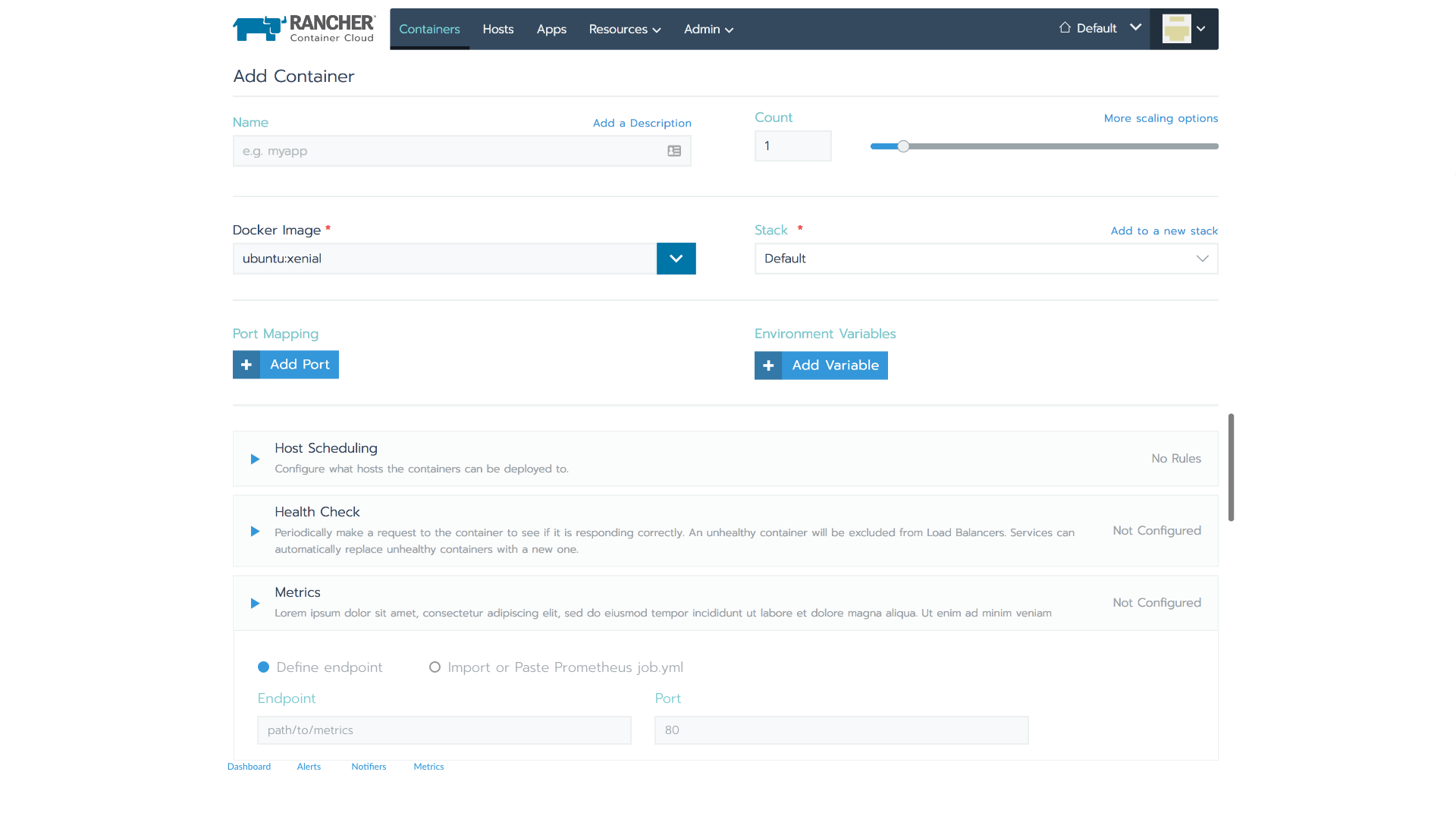
Through a catalog item, we will deploy a managed Prometheus stack. Doing so will allow Rancher UI to show more advanced dashboards with historical data. With the monitoring stack we can provide added value monitoring the Rancher load balancing services.

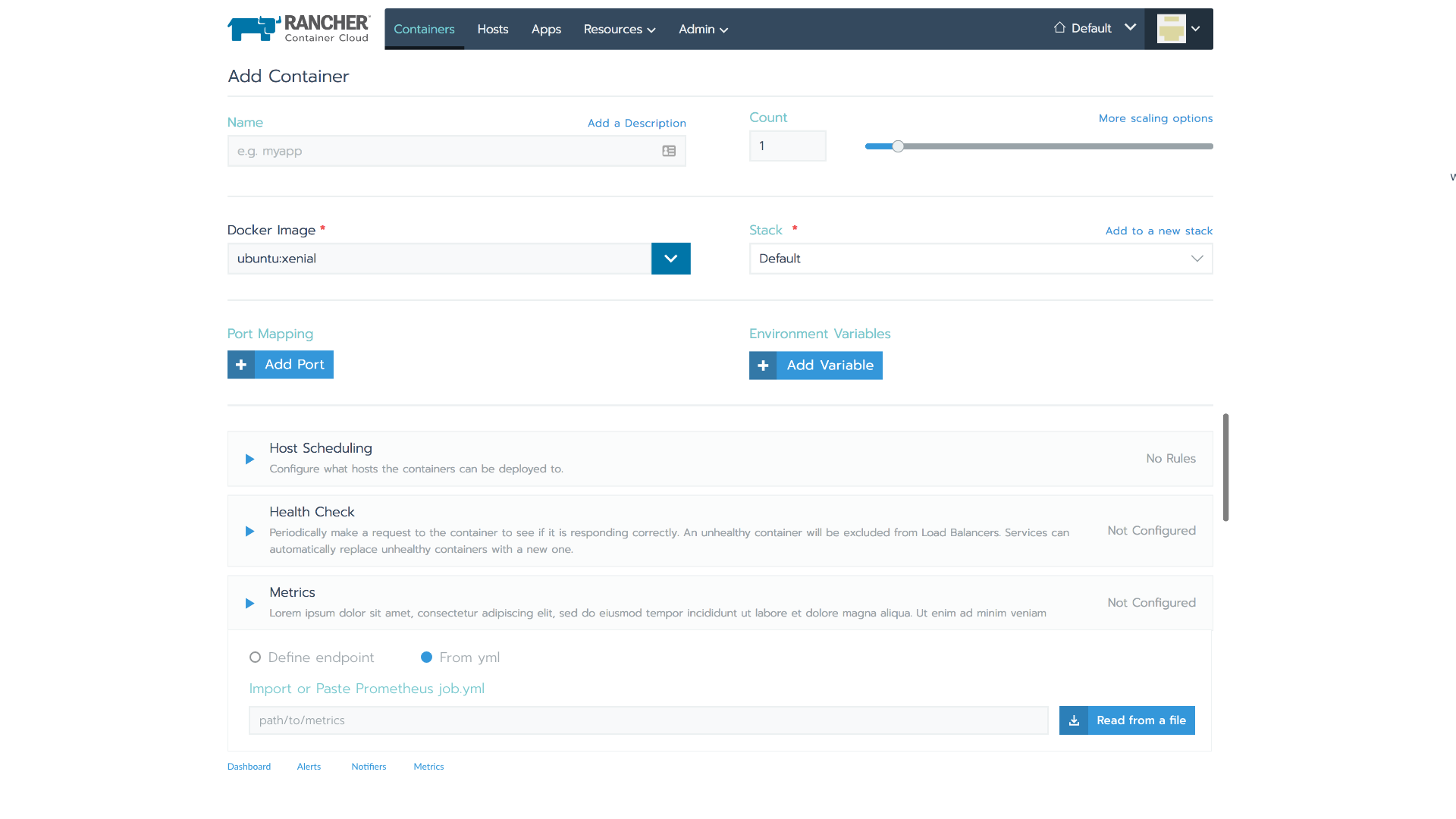


*High level Dashboard*

*Load Balancing Dashboard in Detailed View.*

Once users have deployed the Prometheus stack they will be able to add monitoring endpoints for their services. The option becomes available on the add service/container workflow. Users can add a url and port for simple configurations or provide a complete yaml snippet for a Prometheus job object. There will likely be some work to allow for auto-populating the endpoints as part of Rancher service discovery.





**Implementation:**

**Alert Manager Configuration:**

List Request:

GET ./projects/[:proj\_id]/notifiers

Create Request:

POST ./projects/[:proj\_id]/notifiers Body {notifier}

Get Notifier Request:

GET ./projects/[:proj\_id]/notifiers/[:id]

Update Notifier Request:

POST ./projects/[:proj\_id]/notifiers/[:id] {notifier} (Partials?)

Delete Notifier Request:

DELETE ./projects/[:proj\_id]/notifiers/[:id]

Notifier Object: (Proposed)

notifier:

{

type: “notifier”,

ID: “1externalResourceN”,

kind: “[slack|email|webhook|etc.]”,

slack\_config: { type: slack\_config},

email\_config: {type: email\_config},

webhook\_config: {type: webhook\_config},

pagerduty\_config: {type: pagerduty\_config}

}

Notifier objects will be stored in the Rancher Database. The entire set will need to be sent to any new Alertmanager Operator container that is created. Or it needs to be queriable via API. The config objects will likely have secrets in them and need to be encrypted.

Alerts:

These objects will also be created/observed through the alertmanager operator. The data will be stored in the Rancher database. These alerts will ultimately be translated into alert objects.

Cattle/Rancher Alert objects:

List Request:

GET ./projects/[:proj\_id]/alerts

Response:

Collection of type alert

alert:

{

type: “alert”,

ID: “id”,

labels: {},

annotations: {},

notifier\_id: N

}

**Metadata Observer:**

The metadata observer is responsible for watching Rancher metadata and responding to failing healthchecks based on alerting rules. The metadata observer will need to poll and resend alerts every 30-60 seconds, if not faster. The alert should send the following information to line up with the AlertManager API.

labels:

* Rancher Env
* Stack: nullable
* Service: nullable
* Severity: [Info, Warning, Critical]
* Monitor: (RancherHealthCheckAlerter)
* Alertname: User defined alert name.

Annotations:

* InstanceName
* InstanceID
* Host(agent)

The rules for the monitoring will look like:

{

object: “{service|container|stack|host}”,

condition: “>=”,

value: “X %”,

Action: {

Notify: “{email|webhook|pagerduty|etc.}”

}

}

**Prometheus Configuration:**

**Monitoring/Metrics:**

We will be deploying a managed Prometheus stack that will be pre-wired to monitor Rancher environment level services. Users will be able to add endpoints and the Prometheus operator will pick up the target, and provide the individual container endpoints. There is a basic file level service discovery that can be tapped into initially. If the need arises we can also add a native endpoint into upstream Prometheus. Rancher can also configure and wire up the Kubernetes Prometheus service discovery when necessary.