Perfect ✅ — let’s build a **ready-to-use setup** for Rancher2 → Prometheus → Netcool that:

1. **Separates Infra vs Workload alerts**.
2. **Includes essential cluster-level infra alerts** (nodes, etcd, API server, scheduler, kubelet, networking, disk, memory).
3. Provides a **matching Alertmanager config** + **Netcool probe rules**.

**🔹 Rancher Side Configuration**

**1. Alertmanager Config (separating infra vs workload)**

Create/update alertmanager.yaml inside the secret:

global:

resolve\_timeout: 5m

route:

group\_by: ['alertname', 'namespace', 'pod']

group\_wait: 30s

group\_interval: 5m

repeat\_interval: 1h

routes:

- match:

team: infra

receiver: 'netcool-infra'

- match:

team: workload

receiver: 'netcool-workload'

receivers:

- name: 'netcool-infra'

webhook\_configs:

- url: 'http://netcool-probe.example.com:8080/infra'

send\_resolved: true

- name: 'netcool-workload'

webhook\_configs:

- url: 'http://netcool-probe.example.com:8080/workload'

send\_resolved: true

Apply:

kubectl apply -f alertmanager-secret.yaml

kubectl delete pod -l app.kubernetes.io/name=alertmanager -n cattle-monitoring-system

**2. Infra Alerts (PrometheusRules)**

Deploy cluster-level infra alerts in cattle-monitoring-system namespace.  
Here’s a recommended baseline:

apiVersion: monitoring.coreos.com/v1

kind: PrometheusRule

metadata:

name: infra-alerts

namespace: cattle-monitoring-system

spec:

groups:

- name: infra.rules

rules:

# Node Down

- alert: NodeDown

expr: up{job="kubelet"} == 0

for: 2m

labels:

severity: critical

team: infra

cluster: rancher-prod

annotations:

summary: "Kubernetes node is down"

description: "Node {{ $labels.instance }} has been unreachable for >2m"

# Kubelet Too Many Pods

- alert: KubeletTooManyPods

expr: kubelet\_running\_pod\_count > kube\_node\_status\_allocatable\_pods

for: 5m

labels:

severity: warning

team: infra

cluster: rancher-prod

annotations:

summary: "Kubelet running too many pods"

description: "Node {{ $labels.node }} is running more pods than allocatable"

# API Server Down

- alert: KubeAPIDown

expr: up{job="apiserver"} == 0

for: 1m

labels:

severity: critical

team: infra

cluster: rancher-prod

annotations:

summary: "Kubernetes API server is down"

description: "API server {{ $labels.instance }} is not reachable"

# Scheduler Down

- alert: KubeSchedulerDown

expr: up{job="kube-scheduler"} == 0

for: 1m

labels:

severity: critical

team: infra

cluster: rancher-prod

annotations:

summary: "Kubernetes Scheduler down"

description: "Scheduler is not reachable"

# etcd Member Down

- alert: EtcdMemberDown

expr: etcd\_server\_has\_leader == 0

for: 2m

labels:

severity: critical

team: infra

cluster: rancher-prod

annotations:

summary: "Etcd leader lost"

description: "Etcd cluster has lost its leader"

# Disk Usage High

- alert: NodeDiskUsageHigh

expr: (node\_filesystem\_size\_bytes{fstype!~"tmpfs|overlay"} - node\_filesystem\_free\_bytes{fstype!~"tmpfs|overlay"}) / node\_filesystem\_size\_bytes{fstype!~"tmpfs|overlay"} > 0.9

for: 5m

labels:

severity: warning

team: infra

cluster: rancher-prod

annotations:

summary: "Disk usage > 90%"

description: "Node {{ $labels.instance }} volume {{ $labels.mountpoint }} high usage"

# Node Memory Usage High

- alert: NodeMemoryUsageHigh

expr: (1 - (node\_memory\_MemAvailable\_bytes / node\_memory\_MemTotal\_bytes)) > 0.9

for: 5m

labels:

severity: warning

team: infra

cluster: rancher-prod

annotations:

summary: "Node memory usage > 90%"

description: "Node {{ $labels.instance }} memory usage critical"

**3. Workload Alerts (App Namespace Example)**

Example for app team:

apiVersion: monitoring.coreos.com/v1

kind: PrometheusRule

metadata:

name: workload-alerts

namespace: prod-apps

spec:

groups:

- name: workload.rules

rules:

- alert: AppHighLatency

expr: histogram\_quantile(0.9, rate(http\_request\_duration\_seconds\_bucket{namespace="prod-apps"}[5m])) > 0.5

for: 5m

labels:

severity: warning

team: workload

namespace: prod-apps

cluster: rancher-prod

annotations:

summary: "App latency > 500ms"

description: "App {{ $labels.pod }} has high latency"

**🔹 Netcool Side**

**1. Probe Props File (prometheus.props)**

ProbeName : "Prometheus\_HTTP"

ListenPort : 8080

Transport : http

HTTP.Method : post

HTTP.ContentType : application/json

RulesFile : "prometheus.rules"

MessageLog : "/opt/IBM/tivoli/netcool/omnibus/probes/logs/prometheus.log"

LogFile : "/opt/IBM/tivoli/netcool/omnibus/probes/logs/prometheus\_error.log"

LogLevel : "info"

**2. Probe Rules File (prometheus.rules)**

###############################################################################

# Prometheus Alertmanager → Netcool Probe Rules

###############################################################################

# Parse key JSON fields

JSON PATH "$.alerts[0].labels.alertname" ==> $AlertName

JSON PATH "$.alerts[0].labels.severity" ==> $Severity

JSON PATH "$.alerts[0].annotations.summary" ==> $Summary

JSON PATH "$.alerts[0].annotations.description" ==> $Description

JSON PATH "$.alerts[0].labels.namespace" ==> $Namespace

JSON PATH "$.alerts[0].labels.pod" ==> $Pod

JSON PATH "$.alerts[0].labels.container" ==> $Container

JSON PATH "$.alerts[0].labels.cluster" ==> $Cluster

JSON PATH "$.alerts[0].labels.team" ==> $Team

###############################################################################

# Map to ObjectServer

###############################################################################

@Identifier = $AlertName

@AlertKey = $AlertName:$Namespace:$Pod

@Summary = $Summary

@ExtendedAttr\_Description = $Description

@ExtendedAttr\_Namespace = $Namespace

@ExtendedAttr\_Pod = $Pod

@ExtendedAttr\_Container = $Container

@ExtendedAttr\_Cluster = $Cluster

@ExtendedAttr\_Team = $Team

# Route infra vs workload

if ($Team == "infra") {

@Class = "PrometheusInfra";

@AlertGroup= "Infrastructure";

}

if ($Team == "workload") {

@Class = "PrometheusWorkload";

@AlertGroup= "Applications";

}

# Severity mapping

if ($Severity == "critical") @Severity = 5;

if ($Severity == "warning") @Severity = 4;

if ($Severity == "info") @Severity = 2;

**✅ Workflow**

1. **Infra alerts** (team=infra) → Alertmanager route → Netcool probe /infra → ObjectServer with Class=PrometheusInfra.
2. **Workload alerts** (team=workload) → Alertmanager route → Netcool probe /workload → ObjectServer with Class=PrometheusWorkload.
3. In Netcool UI, build **views/filters** for Infra vs App alerts.

👉 Do you also want me to include a **diagram (flow chart)** that visually shows Rancher Prometheus → Alertmanager → Netcool Probe → ObjectServer (split into Infra vs Workload), so your operations team can include it in SOP docs?

awesome — here’s a **production-ready infra alert pack** (based on the upstream Kubernetes mixins, adapted for Rancher Monitoring v2) plus the matching Alertmanager routing you already asked for. Everything is tagged with team: infra so it’s cleanly separated from user workload alerts going to Netcool.

I’m giving you **three drop-in files**:

1. alertmanager.yaml – routes infra vs workload to different Netcool receivers
2. prometheusrule-infra.yaml – **comprehensive cluster/infra alerts** (CoreDNS, NodeDown, API server, etcd, scheduler, controller-manager, kubelet, kube-proxy, PV/PVC capacity, pod health, daemonsets/deployments/statefulsets, KSM/cAdvisor presence)
3. prometheusrule-workload-sample.yaml – small example showing how app teams label alerts with team: workload (optional)

**1) alertmanager.yaml (Rancher → Netcool separation)**

global:

resolve\_timeout: 5m

route:

group\_by: ['alertname','namespace','pod','node']

group\_wait: 30s

group\_interval: 5m

repeat\_interval: 1h

routes:

- match:

team: infra

receiver: netcool-infra

- match:

team: workload

receiver: netcool-workload

receivers:

- name: netcool-infra

webhook\_configs:

- url: 'http://netcool-probe.example.com:8080/infra'

send\_resolved: true

- name: netcool-workload

webhook\_configs:

- url: 'http://netcool-probe.example.com:8080/workload'

send\_resolved: true

Apply it by updating the Alertmanager secret in cattle-monitoring-system, then restart the Alertmanager pods.

**2) prometheusrule-infra.yaml (Comprehensive infra pack)**

Namespace: **cattle-monitoring-system** (or where Rancher Monitoring is installed)

apiVersion: monitoring.coreos.com/v1

kind: PrometheusRule

metadata:

name: infra-alerts

namespace: cattle-monitoring-system

labels:

app.kubernetes.io/part-of: rancher-monitoring

spec:

groups:

- name: control-plane.health

rules:

# API Server health

- alert: KubeAPIServerDown

expr: up{job=~"apiserver|kube-apiserver"} == 0

for: 2m

labels: {severity: critical, team: infra, category: control-plane}

annotations:

summary: "Kubernetes API server is down"

description: "API server {{ $labels.instance }} is unreachable."

# Scheduler health

- alert: KubeSchedulerDown

expr: up{job=~"kube-scheduler"} == 0

for: 2m

labels: {severity: critical, team: infra, category: control-plane}

annotations:

summary: "Kube-scheduler is down"

description: "Scheduler {{ $labels.instance }} is unreachable."

# Controller-manager health

- alert: KubeControllerManagerDown

expr: up{job=~"kube-controller-manager"} == 0

for: 2m

labels: {severity: critical, team: infra, category: control-plane}

annotations:

summary: "Kube-controller-manager is down"

description: "Controller-manager {{ $labels.instance }} is unreachable."

- name: etcd.health

rules:

# etcd instance down (no scrape)

- alert: EtcdDown

expr: up{job=~"etcd"} == 0

for: 2m

labels: {severity: critical, team: infra, category: etcd}

annotations:

summary: "etcd instance is down"

description: "etcd {{ $labels.instance }} is not responding."

# etcd has no leader

- alert: EtcdNoLeader

expr: etcd\_server\_has\_leader == 0

for: 1m

labels: {severity: critical, team: infra, category: etcd}

annotations:

summary: "etcd cluster has no leader"

description: "etcd cluster lost its leader."

# frequent leader changes

- alert: EtcdHighLeaderChanges

expr: increase(etcd\_server\_leader\_changes\_seen\_total[10m]) > 3

for: 0m

labels: {severity: warning, team: infra, category: etcd}

annotations:

summary: "etcd leader changing frequently"

description: "Leader changed {{ $value }} times in 10m."

# etcd DB size approaching limits (tune threshold to your infra)

- alert: EtcdDatabaseSizeHigh

expr: etcd\_mvcc\_db\_total\_size\_in\_bytes > 3e+09

for: 10m

labels: {severity: warning, team: infra, category: etcd}

annotations:

summary: "etcd database size high"

description: "etcd DB size {{ $value | humanize1024 }} > 3GB."

- name: dns.coredns

rules:

# CoreDNS down (scrape failing)

- alert: CoreDNSDown

expr: up{job=~"coredns"} == 0

for: 2m

labels: {severity: critical, team: infra, category: dns}

annotations:

summary: "CoreDNS is down"

description: "CoreDNS {{ $labels.instance }} is not responding."

# CoreDNS high 99th percentile request latency

- alert: CoreDNSHighRequestLatency

expr: histogram\_quantile(0.99, sum by (le) (rate(coredns\_dns\_request\_duration\_seconds\_bucket[5m]))) > 0.25

for: 10m

labels: {severity: warning, team: infra, category: dns}

annotations:

summary: "CoreDNS high request latency (p99 > 250ms)"

description: "DNS latency elevated for last 10m."

- name: node.health

rules:

# Kubelet down (scrape)

- alert: KubeletDown

expr: up{job=~"kubelet"} == 0

for: 2m

labels: {severity: critical, team: infra, category: node}

annotations:

summary: "Kubelet down"

description: "Kubelet on {{ $labels.instance }} is not responding."

# Node NotReady condition

- alert: NodeNotReady

expr: kube\_node\_status\_condition{condition="Ready",status="true"} == 0

for: 3m

labels: {severity: critical, team: infra, category: node}

annotations:

summary: "Node NotReady"

description: "Node {{ $labels.node }} has been NotReady for >3m."

# DiskPressure / MemoryPressure / PIDPressure

- alert: NodeDiskPressure

expr: kube\_node\_status\_condition{condition="DiskPressure",status="true"} == 1

for: 5m

labels: {severity: warning, team: infra, category: node}

annotations:

summary: "Node DiskPressure"

description: "Node {{ $labels.node }} reports DiskPressure."

- alert: NodeMemoryPressure

expr: kube\_node\_status\_condition{condition="MemoryPressure",status="true"} == 1

for: 5m

labels: {severity: warning, team: infra, category: node}

annotations:

summary: "Node MemoryPressure"

description: "Node {{ $labels.node }} reports MemoryPressure."

- alert: NodePIDPressure

expr: kube\_node\_status\_condition{condition="PIDPressure",status="true"} == 1

for: 5m

labels: {severity: warning, team: infra, category: node}

annotations:

summary: "Node PIDPressure"

description: "Node {{ $labels.node }} reports PIDPressure."

# Node filesystem usage high (exclude ephemeral fs)

- alert: NodeFilesystemUsageHigh

expr: (node\_filesystem\_size\_bytes{fstype!~"tmpfs|overlay"} - node\_filesystem\_free\_bytes{fstype!~"tmpfs|overlay"}) / node\_filesystem\_size\_bytes{fstype!~"tmpfs|overlay"} > 0.9

for: 10m

labels: {severity: warning, team: infra, category: node}

annotations:

summary: "Node filesystem usage > 90%"

description: "Node {{ $labels.instance }} mount {{ $labels.mountpoint }} nearing full."

# Node memory high

- alert: NodeMemoryUsageHigh

expr: (1 - (node\_memory\_MemAvailable\_bytes / node\_memory\_MemTotal\_bytes)) > 0.9

for: 10m

labels: {severity: warning, team: infra, category: node}

annotations:

summary: "Node memory usage > 90%"

description: "Node {{ $labels.instance }} memory pressure."

- name: kube.components

rules:

# kube-proxy down

- alert: KubeProxyDown

expr: up{job=~"kube-proxy"} == 0

for: 3m

labels: {severity: warning, team: infra, category: networking}

annotations:

summary: "kube-proxy down"

description: "kube-proxy on {{ $labels.instance }} not scraping."

# kube-state-metrics down

- alert: KubeStateMetricsDown

expr: up{job=~"kube-state-metrics"} == 0

for: 2m

labels: {severity: warning, team: infra, category: monitoring}

annotations:

summary: "kube-state-metrics down"

description: "KSM {{ $labels.instance }} not responding."

# node-exporter down

- alert: NodeExporterDown

expr: up{job=~"node-exporter"} == 0

for: 2m

labels: {severity: warning, team: infra, category: monitoring}

annotations:

summary: "node-exporter down"

description: "node-exporter {{ $labels.instance }} not responding."

# cAdvisor metrics missing (via kubelet)

- alert: KubeletCAdvisorMetricsMissing

expr: absent(container\_cpu\_usage\_seconds\_total)

for: 10m

labels: {severity: warning, team: infra, category: monitoring}

annotations:

summary: "cAdvisor metrics missing"

description: "cAdvisor container metrics absent for 10m+."

- name: workloads.platform\_safety

rules:

# CrashLooping containers (across cluster)

- alert: ContainerCrashLooping

expr: increase(kube\_pod\_container\_status\_restarts\_total[5m]) > 3

for: 10m

labels: {severity: warning, team: infra, category: workloads}

annotations:

summary: "Pods restarting frequently"

description: "Container {{ $labels.container }} in pod {{ $labels.pod }} is restarting >3 times/5m."

# Pending pods (cluster resource pressure / scheduling)

- alert: PodPendingTooLong

expr: max by (namespace) (kube\_pod\_status\_phase{phase="Pending"}) > 0

for: 15m

labels: {severity: warning, team: infra, category: scheduling}

annotations:

summary: "Pods pending > 15m"

description: "Pending pods detected. Check scheduler/resources."

# Image pull errors

- alert: ImagePullBackOff

expr: sum by (namespace,pod) (kube\_pod\_container\_status\_waiting\_reason{reason="ImagePullBackOff"}) > 0

for: 10m

labels: {severity: warning, team: infra, category: workloads}

annotations:

summary: "ImagePullBackOff"

description: "Pod {{ $labels.pod }} is failing to pull image."

# OOM killed containers

- alert: ContainerOOMKilled

expr: increase(kube\_pod\_container\_status\_terminated\_reason{reason="OOMKilled"}[10m]) > 0

for: 0m

labels: {severity: warning, team: infra, category: workloads}

annotations:

summary: "Container OOMKilled"

description: "Container {{ $labels.container }} in pod {{ $labels.pod }} OOMKilled."

- name: controllers.readiness

rules:

# Deployment not enough available replicas

- alert: DeploymentReplicasNotAvailable

expr: (kube\_deployment\_status\_replicas\_available < kube\_deployment\_spec\_replicas)

for: 10m

labels: {severity: warning, team: infra, category: controllers}

annotations:

summary: "Deployment unavailable replicas"

description: "Deployment {{ $labels.deployment }} not fully available."

# DaemonSet unavailable

- alert: DaemonSetUnavailable

expr: kube\_daemonset\_status\_number\_unavailable > 0

for: 10m

labels: {severity: warning, team: infra, category: controllers}

annotations:

summary: "DaemonSet pods unavailable"

description: "DaemonSet {{ $labels.daemonset }} has unavailable pods."

# StatefulSet not ready

- alert: StatefulSetReplicasNotReady

expr: kube\_statefulset\_status\_replicas\_ready < kube\_statefulset\_replicas

for: 10m

labels: {severity: warning, team: infra, category: controllers}

annotations:

summary: "StatefulSet replicas not ready"

description: "StatefulSet {{ $labels.statefulset }} not fully ready."

# Job failures

- alert: JobFailed

expr: increase(kube\_job\_status\_failed[10m]) > 0

for: 0m

labels: {severity: warning, team: infra, category: controllers}

annotations:

summary: "Kubernetes job failed"

description: "Job {{ $labels.job\_name }} encountered failure(s)."

- name: storage.capacity

rules:

# PVC pending (no PV bound)

- alert: PVCPending

expr: kube\_persistentvolumeclaim\_status\_phase{phase="Pending"} == 1

for: 10m

labels: {severity: warning, team: infra, category: storage}

annotations:

summary: "PVC Pending"

description: "PVC {{ $labels.persistentvolumeclaim }} pending >10m."

# PV not bound / failed

- alert: PVNotBoundOrFailed

expr: kube\_persistentvolume\_status\_phase{phase!~"Bound|Available"} == 1

for: 10m

labels: {severity: warning, team: infra, category: storage}

annotations:

summary: "PV not in healthy phase"

description: "PV {{ $labels.persistentvolume }} in phase {{ $labels.phase }}."

# Volume free space low (via kubelet volume stats)

- alert: VolumeSpaceLow

expr: (kubelet\_volume\_stats\_available\_bytes / kubelet\_volume\_stats\_capacity\_bytes) < 0.1

for: 15m

labels: {severity: warning, team: infra, category: storage}

annotations:

summary: "Volume free space < 10%"

description: "PVC {{ $labels.persistentvolumeclaim }} low free space."

# Volume inodes low

- alert: VolumeInodesLow

expr: (kubelet\_volume\_stats\_inodes\_free / kubelet\_volume\_stats\_inodes) < 0.1

for: 15m

labels: {severity: warning, team: infra, category: storage}

annotations:

summary: "Volume inodes < 10% free"

description: "PVC {{ $labels.persistentvolumeclaim }} low free inodes."

Notes

* These rules expect **kube-state-metrics**, **node-exporter**, **kubelet/cAdvisor** and standard component scrape jobs enabled by Rancher Monitoring v2.
* Tune thresholds (latency, space %, time windows) to your SLOs.
* Everything is tagged team: infra, which drives Alertmanager → Netcool “infra” receiver.

**3) prometheusrule-workload-sample.yaml (optional pattern for app teams)**

apiVersion: monitoring.coreos.com/v1

kind: PrometheusRule

metadata:

name: workload-alerts

namespace: my-business-namespace

spec:

groups:

- name: app.rules

rules:

- alert: HighAppLatencyP90

expr: histogram\_quantile(0.9, rate(http\_request\_duration\_seconds\_bucket{namespace="my-business-namespace"}[5m])) > 0.5

for: 5m

labels:

severity: warning

team: workload

namespace: my-business-namespace

annotations:

summary: "App latency > 500ms (p90)"

description: "Pod {{ $labels.pod }} is over latency threshold."

**Netcool side (recap)**

Your **probe rules** should already parse team and split classes:

JSON PATH "$.alerts[0].labels.team" ==> $Team

JSON PATH "$.alerts[0].labels.severity" ==> $Severity

JSON PATH "$.alerts[0].annotations.summary" ==> $Summary

JSON PATH "$.alerts[0].annotations.description" ==> $Description

JSON PATH "$.alerts[0].labels.alertname" ==> $AlertName

JSON PATH "$.alerts[0].labels.namespace" ==> $Namespace

JSON PATH "$.alerts[0].labels.pod" ==> $Pod

JSON PATH "$.alerts[0].labels.node" ==> $Node

@Identifier = $AlertName

@AlertKey = $AlertName:$Namespace:$Pod

@Summary = $Summary

@Node = $Node

@ExtendedAttr\_Namespace = $Namespace

@ExtendedAttr\_Pod = $Pod

@ExtendedAttr\_Team = $Team

if ($Team == "infra") { @Class="PrometheusInfra"; @AlertGroup="Infrastructure"; }

if ($Team == "workload") { @Class="PrometheusWorkload"; @AlertGroup="Applications"; }

if ($Severity == "critical") @Severity=5;

if ($Severity == "warning") @Severity=4;

if ($Severity == "info") @Severity=2;

**What to customize quickly**

* Replace any cluster/name labels you want propagated (e.g., add cluster: rancher-prod to rule labels so Netcool sees it).
* If your scrape job names differ (e.g., job="apiserver" vs job="kube-apiserver"), keep the =~ alternations or adjust to your environment.
* For very large clusters, consider **per-namespace rate limits** or tune group\_by to reduce cardinality.

If you want, I can also give you a **one-liner kubectl patch** that injects team: infra automatically into Rancher’s built-in mixin rules, but the cleanest approach is to manage your own PrometheusRule CRDs (like above) so they’re version-controlled.

Perfect 👍 Let’s build a **complete set** for you.  
We’ll cover both sides: **Rancher Prometheus Rules** (cluster infra alerts) **+ Netcool probe rules** (with ABC Team mapping).

**🔹 Rancher Side – Prometheus Rules (Infra Alerts)**

We’ll include **critical infra alerts**:

* NodeDown
* CoreDNSDown
* KubeAPIServerDown
* etcdHighLatency
* PodCrashLoopBackOff (system namespaces)
* KubeletTooManyPods

groups:

- name: infra-alerts

rules:

# Node down

- alert: NodeDown

expr: up{job="kubelet"} == 0

for: 2m

labels:

severity: critical

team: infra

cluster: rancher-prod

annotations:

summary: "Node is down"

description: "Kubelet on {{ $labels.instance }} is not responding"

node: "{{ $labels.instance }}"

# CoreDNS pods not running

- alert: CoreDNSDown

expr: kube\_deployment\_status\_replicas\_available{deployment="coredns", namespace="kube-system"} < 2

for: 5m

labels:

severity: critical

team: infra

cluster: rancher-prod

annotations:

summary: "CoreDNS unavailable"

description: "CoreDNS deployment in kube-system is below required replicas"

namespace: "kube-system"

# API Server down

- alert: KubeAPIServerDown

expr: up{job="apiserver"} == 0

for: 2m

labels:

severity: critical

team: infra

cluster: rancher-prod

annotations:

summary: "Kubernetes API server is down"

description: "API server is not reachable"

namespace: "kube-system"

# etcd latency

- alert: EtcdHighLatency

expr: histogram\_quantile(0.99, rate(etcd\_request\_duration\_seconds\_bucket[5m])) > 0.25

for: 5m

labels:

severity: warning

team: infra

cluster: rancher-prod

annotations:

summary: "etcd high latency"

description: "etcd has >250ms p99 latency"

namespace: "kube-system"

# CrashLoop in system pods

- alert: PodCrashLoopBackOff

expr: kube\_pod\_container\_status\_waiting\_reason{reason="CrashLoopBackOff", namespace=~"kube-system|monitoring"} > 0

for: 10m

labels:

severity: critical

team: infra

cluster: rancher-prod

annotations:

summary: "Pod CrashLoopBackOff detected"

description: "Pod {{ $labels.pod }} in {{ $labels.namespace }} is crash looping"

namespace: "{{ $labels.namespace }}"

pod: "{{ $labels.pod }}"

node: "{{ $labels.node }}"

# Node overloaded with pods

- alert: KubeletTooManyPods

expr: kubelet\_running\_pod\_count > 250

for: 5m

labels:

severity: warning

team: infra

cluster: rancher-prod

annotations:

summary: "Node too many pods"

description: "Node {{ $labels.node }} has more than 250 pods running"

node: "{{ $labels.node }}"

👉 All of these have team: infra and **annotations with summary, description, namespace, pod, node, cluster**.  
These go to Alertmanager → Netcool Probe webhook.

**🔹 Netcool Side – Probe Rules (prometheus.rules)**

Here’s a **best-practice rule file** tuned for ABC Team.  
It parses severity, summary, description, alertname, namespace, pod, node, cluster, and team.

# Extract base fields from Prometheus webhook

JSON PATH "$.alerts[0].labels.alertname" ==> $AlertName

JSON PATH "$.alerts[0].labels.severity" ==> $Severity

JSON PATH "$.alerts[0].labels.team" ==> $Team

JSON PATH "$.alerts[0].labels.cluster" ==> $Cluster

JSON PATH "$.alerts[0].labels.namespace" ==> $Namespace

JSON PATH "$.alerts[0].labels.pod" ==> $Pod

JSON PATH "$.alerts[0].labels.node" ==> $Node

JSON PATH "$.alerts[0].annotations.summary" ==> $Summary

JSON PATH "$.alerts[0].annotations.description" ==> $Description

# Map into Netcool event fields

@Class = "PrometheusInfra"

@Identifier = $AlertName

@Node = $Node

@Manager = "Prometheus-Cluster"

@Agent = "RancherPrometheus"

@Severity = @SeverityFromLabel($Severity)

@Summary = $Summary

@Description = $Description

@AlertGroup = "Infrastructure"

@Cluster = $Cluster

@Namespace = $Namespace

@Pod = $Pod

@Owner = "ABC Team"

# Convert severity label into ObjectServer numeric severity

@SeverityFromLabel($label) {

if ($label == "critical") return 5; # Critical

if ($label == "warning") return 3; # Minor

if ($label == "info") return 1; # Information

return 2; # Default to indeterminate

}

**🔹 How this works in Netcool**

* **Class**: PrometheusInfra → all infra alerts are grouped.
* **Owner**: ABC Team → clearly shows who owns the alert.
* **Severity mapping**:
  + critical → Netcool severity 5
  + warning → Netcool severity 3
  + info → Netcool severity 1
* **Summary/Description**: Passed directly from annotations.
* **Node, Pod, Namespace**: Available for filtering.
* **Cluster**: Useful if you monitor multiple Rancher clusters.

**✅ Result**

* Rancher → Prometheus rules generate **infra alerts**.
* Alertmanager routes to Netcool probe.
* Probe parses → ObjectServer events with:
  + Class=PrometheusInfra
  + Owner=ABC Team
  + Severity based on Prometheus labels
  + Rich context (summary, description, namespace, pod, node, cluster)

👉 Do you also want me to create a **matching Alertmanager config** that **routes only team=infra alerts** to this Netcool probe (and leaves room for workload team later)?