Preparing for Monitoring

Prepare the platform

Stop the Codeready Containers machine, enable monitoring and restart the machine with 16GB memory.

```
$ crc stop
$ crc config set enable-cluster-monitoring true
$ crc start -m 16384
```

The startup should take 5-10 minutes.

Configure monitoring for the platform and for user projects

Monitoring configuration is centralized in two config maps, one for platform monitoring, one for user workloads. First check if these config maps exist (initially, they should not).

```
$ oc -n openshift-monitoring get configmap cluster-monitoring-config
$ oc -n openshift-user-workload-monitoring get configmap user-workload-monitoring-config
```

If they don't exist, create them.

The manifest for platform monitoring is named `cluster-monitoring-config.yaml`. It enables user workload monitoring, configures a retention period and requests storage for Prometheus.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    enableUserWorkload: true
    prometheusK8s:
      retention: 12h
      volumeClaimTemplate:
       spec:
         volumeMode: Filesystem
         resources:
           requests:
             storage: 5Gi
```

The manifest for user workload monitoring is named `user-monitoring-config.yaml`. It configures a retention period and requests CPU and RAM for Prometheus.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheus:
      retention: 12h
      resources:
        requests:
          cpu: 200m
          memory: 2Gi
```

Note that the config maps reside in different projects.

Apply the manifests. After a short time, check if the required pods for monitoring user projects run.

```
$ oc apply -f .
$ oc -n openshift-user-workload-monitoring get pod
NAME                                   READY   STATUS    RESTARTS   AGE
prometheus-operator-85f5687dd8-8frfm   2/2     Running   0          3m41s
prometheus-user-workload-0             5/5     Running   1          3m37s
thanos-ruler-user-workload-0           3/3     Running   0          3m34s
```

Fix monitoring

Unfortunately, the configuration won't work. Prometheus doesn't seem to publish any samples. Troubleshoot this.

```
$ oc login kubeadmin
$ oc project openshift-monitoring
$ oc get pod
```

The Prometheus pod is not running.

```
$ oc describe PROMETHEUS_POD | more
```

When you scroll through the text, try to detect error messages. You should find "permissions denied" on a subdirectory of /prometheus.

This directory is listed in the volume mounts. It's the PVC that you put into the config map. Remove it, apply it again and after 30 seconds, the Prometheus pod should be up and running. You can use `cluster-monitoring-config-corrected.yaml`.

Use RBAC to give users monitoring privileges

Give *developer* permission to view and manipulate monitoring rules in *devproject*.

```
$ oc policy add-role-to-user monitoring-rules-view developer -n devproject
$ oc policy add-role-to-user monitoring-rules-edit developer -n devproject
```

Enter the web console as *kubeadmin* and check whether this was successful.

Obtain the location of the Prometheus API

You need a token to authenticate with Prometheus, and the name of the host where the Prometheus API runs. The token is stored in a secret, and the host is in a route in the user workload monitoring project.

The resources below reside in administrative projects `openshift-monitoring` and `openshift-user-workload-monitoring`. Normal users can't see them unless a cluster admin gives them permission, e.g. with the *view* role (security problem). Alternatively, users can receive the API host and the token from the cluster admin.

Ensure you are *kubeadmin*. Obtain the API host.

```
$ oc describe route thanos-querier -n openshift-monitoring
```

The host is listed as *Requested Host* and should be *thanos-querier-openshift-monitoring.apps-crc.testing*.

Obtain the Prometheus API token

List the monitoring secrets and limit the list to those named prometheus-user-workload-token. Pick one and get its token.

```
$ oc get secret -n openshift-user-workload-monitoring
prometheus-user-workload-token-c2vhf           kubernetes.io/service-account-token   4      41m
prometheus-user-workload-token-lt7ht           kubernetes.io/service-account-token   4      41m
$ oc describe secret prometheus-user-workload-token-c2vhf -n openshift-user-workload-monitoring
...
token:           eyJhbGciOiJSUzI1NiIsImtpZCI6IlJmSlQ3Qk9EcUJLYW ...
```

The token is fairly long. Copy it and paste it in a shell variable.

```
$ T=eyJhbGciOiJSUzI1NiIsImtpZCI6IlJmSlQ3Qk9EcUJLYW ...
```

Make a test query.

```
$ curl -X GET -kG "https://thanos-querier-openshift-monitoring.apps-crc.testing/api/v1/query?" \
         --data-urlencode "query=up{namespace='devproject'}" -H "Authorization: Bearer $T"
```

This query returns the uptimes of your application pods. If no pods are running, you will receive a successful, but empty response. Launch an app and try again in ten minutes.

Monitor an application

Launch an application

Any application is fine, for example one of the applications deployed during the OpenShift building module (parksmap, django blog or the nodejs server), or the NGINX template. Note down the application's name.

In addition, deploy the app in the sampe-app.yaml manifest. This application accesses Prometheus itself.

```
$ oc apply -f sample-app.yaml
```

View metrics

Log on to the web console as *developer*. Go to the **monitoring** section and explore it.

Dashboard and **Metrics** provide information of the pods' CPU, memory and network usage. When clicking on the Inspect link in the dashboard graphs, the corresponding metrics are displayed.

Metrics provides more detail and allows you to write custom queries in the Prometheus query language PromQL.

Alerts display any alerts.

Events show pod lifecycle events similarly to the `oc describe pod` output.

Cause CPU load

Launch a shell in a pod and run this loop:

```
$ while true; do true; done
```

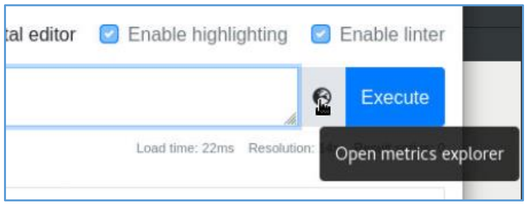
View the effect on the CPU metric in the web console. View both the Dashboard and the Metrics section.

Access the Prometheus dashboard

As kubeadmin, find the prometheus route.

```
$ oc get routes -n openshift-monitoring | grep prometheus-k8s
prometheus-k8s      prometheus-k8s-openshift-monitoring.apps-crc.testing      prometheus-k8s   ...
```

The orange text is Prometheus' endpoint. To view the browser, add the `/graph` path at the end and `http://` at the beginning and enter it in the browser. The Prometheus interface features a metrics explorer:



You could use it to craft alert rules, but you need to know the Prometheus query language.

Create an alert

Manifest `alert.yaml` triggers an alert when pods with the label "*name=testapp*" use more than 50% CPU. Replace this label with a label of your high-CPU pod and apply the alert.

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  namespace: devproject
  name: testapp-cpu-rule
spec:
  groups:
  - name: testapp-high-cpu-rule
    rules:
    - alert: pod_high_cpu
      expr: pod:container_cpu_usage:sum{name="testapp"}>50
      labels:
        severity: critical
```

An alternative manifest is `alert2.yaml`. It triggers an alert when an deploymentconfig with the name testapp has a CPU usage of over 50%. Feel free to try it after adapting the workload name to your apps.

Enter the webconsole as developer and view alerts under the **Monitoring** menu. It will take a few seconds until you see a result.

Monitoring from the cluster admin perspective

As *kubeadmin*, log on to the web console and enter the Metrics section. The data you see is related to the cluster, not any project.

Are there any alerts? Which?

Go to Dashboards and view the graphs that are available.

Select the Dashboard *Kubernetes/Compute Resources/Workload*, Namespace *devproject* and Workload testapp. Click *Inspect* to view the corresponding Metric page.