Table of Contents

* [Rocket.Chat Alerts Lab](https://www.opentlc.com/labs/ocp_operations/03_02_Alerting_with_RocketChat_Lab.html#_rocket_chat_alerts_lab)
* [1. Set Up Rocket.Chat](https://www.opentlc.com/labs/ocp_operations/03_02_Alerting_with_RocketChat_Lab.html#_set_up_rocket_chat)
* [2. Configure Rocket.Chat](https://www.opentlc.com/labs/ocp_operations/03_02_Alerting_with_RocketChat_Lab.html#_configure_rocket_chat)
  + [2.1. Connect to Rocket.Chat and Register Account](https://www.opentlc.com/labs/ocp_operations/03_02_Alerting_with_RocketChat_Lab.html#_connect_to_rocket_chat_and_register_account)
  + [2.2. Create Channel in Rocket.Chat](https://www.opentlc.com/labs/ocp_operations/03_02_Alerting_with_RocketChat_Lab.html#_create_channel_in_rocket_chat)
  + [2.3. Create Inbound Webhook](https://www.opentlc.com/labs/ocp_operations/03_02_Alerting_with_RocketChat_Lab.html#_create_inbound_webhook)
* [3. Install Alertmanager](https://www.opentlc.com/labs/ocp_operations/03_02_Alerting_with_RocketChat_Lab.html#_install_alertmanager)
* [4. Test Alerting](https://www.opentlc.com/labs/ocp_operations/03_02_Alerting_with_RocketChat_Lab.html#_test_alerting)
* [5. Clean Up](https://www.opentlc.com/labs/ocp_operations/03_02_Alerting_with_RocketChat_Lab.html#_clean_up)

Rocket.Chat Alerts Lab

Rocket.Chat is an open source chat platform similar to Slack. You can easily deploy it on top of OpenShift and use it as an example of how to receive alerts from Prometheus in case things go wrong.

1. Set Up Rocket.Chat

1. Log in again to the shell on your Bastion host and switch to the **root** user.
2. Create a new project to hold the Rocket.Chat instance:

oc new-project rocketchat --display-name="Rocket Chat"

1. Deploy a persistent MongoDB database to hold the Rocket.Chat configuration:

oc new-app mongodb-persistent -p MONGODB\_USER=rocketchat-admin -p MONGODB\_PASSWORD=rocketchat -p MONGODB\_DATABASE=rocketchat -p VOLUME\_CAPACITY=4Gi

1. Wait until the MongoDB pod is fully initialized.
2. Deploy the Rocket Chat image from Docker Hub, connecting it to the MongoDB database that you just created, and expose the Rocket Chat service as a route:
3. oc new-app docker.io/rocketchat/rocket.chat:0.59.1 -e MONGO\_URL=mongodb://rocketchat-admin:rocketchat@mongodb:27017/rocketchat

oc expose svc rocketchat

1. Monitor your pods and wait until Rocket Chat fully starts.
   * Tail the pod logs to make sure it starts successfully.
2. Open a **rsh** session into your MongoDB pod:

oc rsh <mongodb\_pod>

|  |  |
| --- | --- |
|  | Use **oc get pod** to find the specific name of your MondoDB pod—expect it to be similar to **mongodb-1-mgvb2**. |

1. Once inside your MongoDB pod, connect to the database:

mongo localhost:27017

1. Enter the following commands one by one:
2. use rocketchat
3. db.auth('rocketchat-admin','rocketchat')
4. db.rocketchat\_settings.update({\_id:'Accounts\_UseDNSDomainCheck'},{$set:{value:false}})

exit

* + Rocket.Chat uses a domain check code to verify the validity of the email address. Because you do not have any email providers available in the lab, you disable this check.

1. Type **exit** and then press **Enter** to disconnect from your MondoDB pod.

2. Configure Rocket.Chat

2.1. Connect to Rocket.Chat and Register Account

In this section, you connect to Rocket.Chat and register a new account.

1. Determine the URL of your Rocket.Chat instance:

oc get route rocketchat

1. In a web browser, navigate to the Rocket.Chat URL.
2. Under the blue **Login** button, select **Register a new account**.
3. Enter your details (Name, e-mail address and pick a password).
4. Click **Register A New Account**.
5. On the warning dialog, click **Yes**.
6. When prompted to register the username, update the suggestion or keep the default, and click **Use this username**.

2.2. Create Channel in Rocket.Chat

In this section, you create a channel in Rocket.Chat so that Alertmanager has a destination for sending alerts.

1. Next to the **Search** box under your name click **+**.
2. Use **openshift-alerts** as the channel name and click **Create**.

2.3. Create Inbound Webhook

In this section, you create the inbound webhook for the new channel to receive alerts from Alertmanager.

1. Next to your username, click the three dots and select **Administration**.
2. Select **Integrations**, then **New Integration**, and then **Incoming WebHook**.
3. Enter or select the following values:
   * **Enabled**: **True**
   * **Name**: **OpenShift Alerts**
   * **Post to Channel**: **#openshift-alerts**
   * **Post as**: **rocket.cat**
   * **Alias**: **Prometheus**
   * **Script Enabled**: **True**
4. Copy and paste the script below into the **Script** box:
5. class Script {
6. process\_incoming\_request({
7. request
8. }) {
9. var alertColor = "warning";
10. var alertKind = "UNDEFINED";
11. if (request.content.status == "resolved") {
12. alertColor = "good";
13. alertKind = "RESOLVED"
14. } else if (request.content.status == "firing") {
15. alertColor = "danger";
16. alertKind = "PROBLEM"
17. }
18. console.log(request.content);
19. let finFields = [];
20. for (i = 0; i < request.content.alerts.length; i++) {
21. var endVal = request.content.alerts[i];
22. var elem = {
23. title: alertKind + ": " + endVal.labels.alertname,
24. value: "Instance: " + endVal.labels.instance + "; " + "Description: " + endVal.annotations.description + "; " + "Summary: " + endVal.annotations.summary,
25. short: false
26. };
27. finFields.push(elem);
28. }
29. return {
30. content: {
31. username: "Alertmanager",
32. attachments: [{
33. color: alertColor,
34. title\_link: request.content.externalURL,
35. title: "Prometheus notification",
36. fields: finFields,
37. }]
38. }
39. };
40. return {
41. error: {
42. success: false,
43. message: 'Error accepting Web Hook'
44. }
45. };
46. }

}

1. Click **Save Changes**.
2. Copy the webhook URL—you need it later to configure Alertmanager.
   * This field appears only after clicking **Save Changes**.
3. Next to **Administration**, click the **x** and return to the chat window.
4. From a shell prompt, test that your webhook is working by sending an example JSON payload to Rocket.Chat, replacing **<WEBHOOK\_URL>** with the URL you copied down earlier:

curl -X POST -H 'Content-Type: application/json' --data '{"alerts": [{"status": "testing", "labels": {"alertname": "alert\_test", "instance": "instance.my.test.cluster" }, "annotations": { "description": "Alert Test Description", "summary": "Alert Test Summary" } }]}' <WEBHOOK\_URL>

* + Expect to receive an alert message in your Rocket.Chat channel.

3. Install Alertmanager

Now that you have a target to receive your alerts, you configure Alertmanager to send alerts when things go wrong.

1. On your Bastion host, change to the directory with the Alertmanager code and switch projects to the **prometheus** OpenShift project:
2. cd /root/openshift-prometheus/alertmanager

oc project prometheus

1. Create Alertmanager, replacing **<WEBHOOK\_URL>** with the previously copied webhook URL:

oc new-app -f alertmanager.yaml -p VOLUME\_CAPACITY=4Gi -p WEBHOOK\_URL=<WEBHOOK\_URL>

1. You can see the configuration of Alertmanager by examining the configuration map created by the template:

oc get cm alertmanager -o yaml

1. Examine the end of the file to see the section that configures Alertmanager to send the alert to a webhook:
2. receivers:
3. - name: 'webhook'
4. webhook\_configs:
5. - send\_resolved: true

url: '<WEBHOOK\_URL'

|  |  |
| --- | --- |
|  | If you want to use other notification mechanisms such as email, PagerDuty, or Slack, you update the configuration map with the correct configuration, as documented in <https://prometheus.io/docs/alerting/configuration>). |

4. Test Alerting

In this section, you test the alerting to make sure it works.

1. From your Bastion host, remotely connect using SSH into one of the nodes.
2. Shut down the **atomic-openshift-node** service:

systemctl stop atomic-openshift-node

|  |  |
| --- | --- |
|  | You may want to make sure that this node is not running either Alertmanager or the Prometheus pod. You can use **oc get pod -n project -o wide** to display which pods run on which nodes. Your best bet probably is to shut down an infranode rather than a worker node, because the Rocket.Chat and Prometheus pods are usually distributed to regular nodes. |

* + After a minute or two, expect to see an alert with a red bar and some descriptive text in the Rocket.Chat window.
  + You can also see the alert in both the Prometheus and Alertmanager user interfaces.

1. Bring the node back online:

systemctl start atomic-openshift-node

* + A few minutes later, expect to see a second notification with the same text, but with a green bar indicating that the situation is resolved.

5. Clean Up

1. If you want to remove your monitoring environment, delete the **prometheus** OpenShift project and the **prometheus-cluster-reader** cluster role-binding:
2. oc delete project prometheus

oc delete clusterrolebinding prometheus-cluster-reader

Build Version: be32934b41da1e34c27e21e79bdf57d30c732a3e : Last updated 2018-01-19 05:59:04 EST