Note: For a summary and decision, scroll to the last page.

**Azure Alerts Overview**

**Basic Azure Alerts Info:**

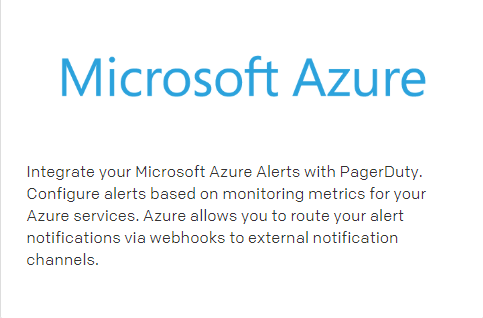
With Azure Monitoring Services, alerts can monitor metrics and log data. Multiple types of Azure resource monitors can be consolidated in a single interface. The resource monitors include but are not limited to PaaS Resource Metrics/Logs, Azure Activity Events, and Log Analytics.

**Metric Alerts:** Examples include *Average\_Avg. Disk sec/Read, Average\_% Processor Time, etc.*

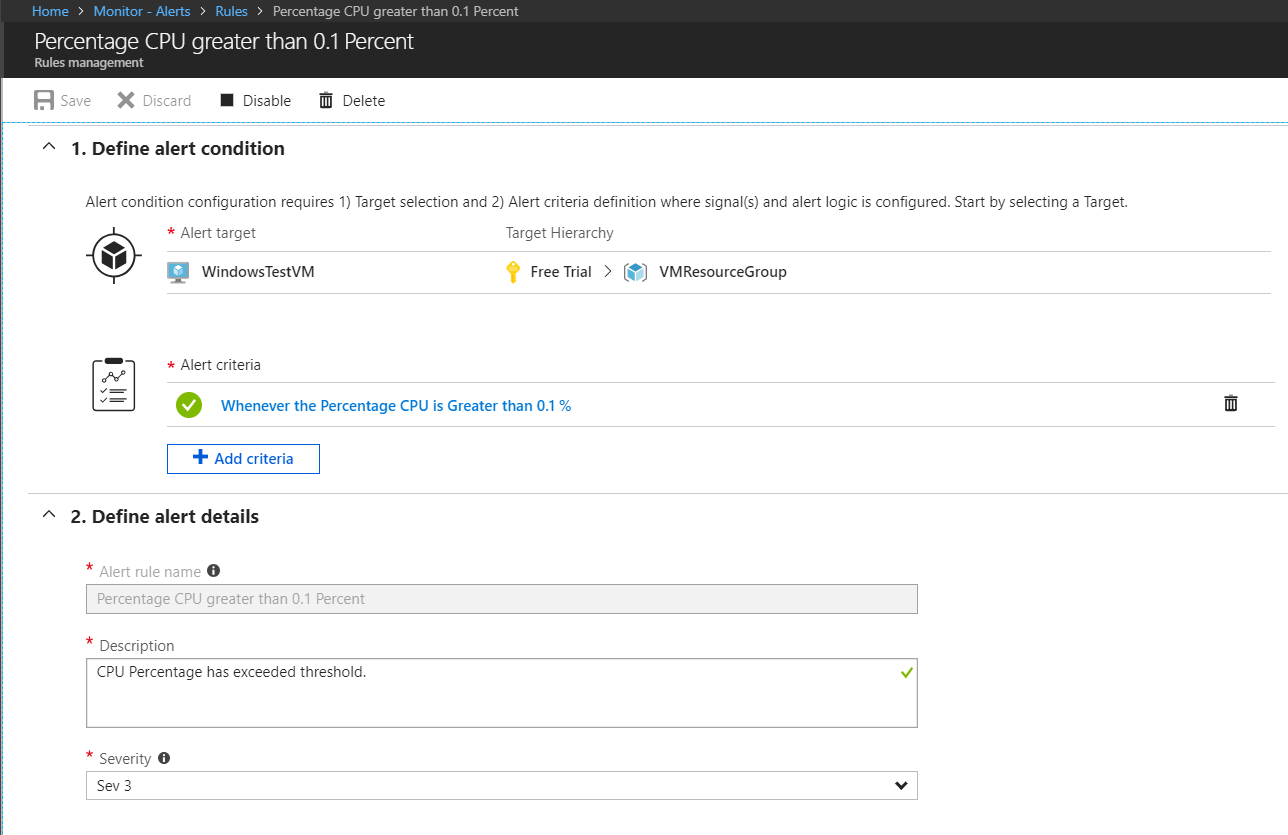
**Log Alerts:** Consists of Log Search rules for Azure Log Analytics or Application Insights. Example:

* Query: requests | where resultCode == "500"
* Time period: 30 minutes
* Alert frequency: five minutes
* Threshold value: Greater than 0

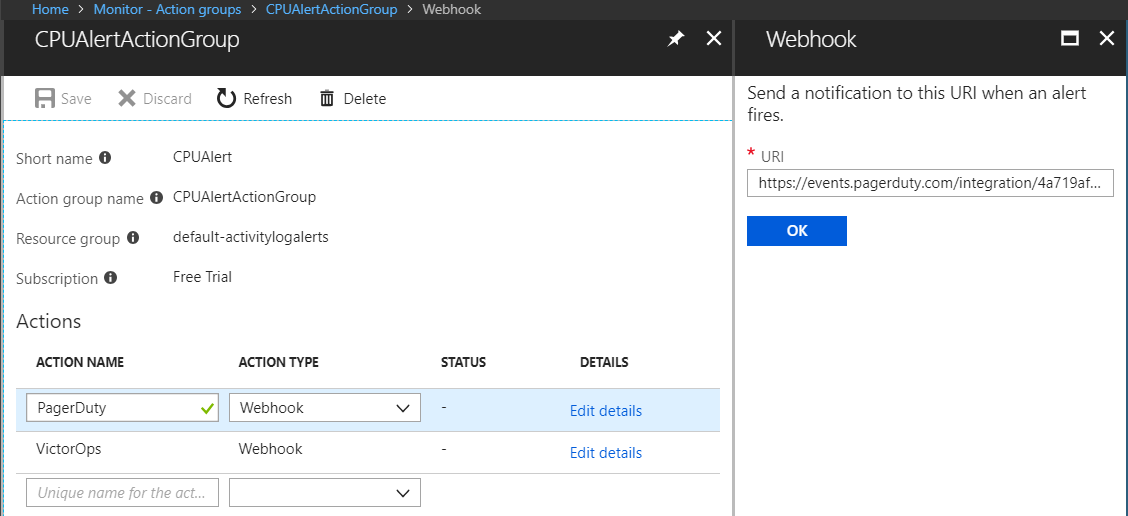
**Example Azure Alert:**

****

*Webhooks* are simply HTTP calls made by Azure to our incident management platform of choice. They notify the correct channel (a designated integration URL) with the original alert information from Azure.



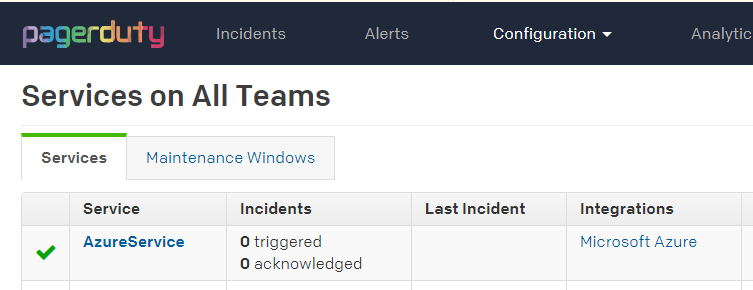
As an example, we set up a test alert that triggers when the CPU usage rises above 0.1%.



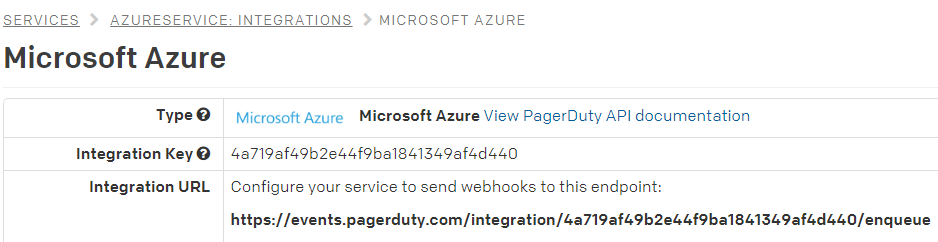
Here we set up an action group that will be used by the alerts to make a webhook with PagerDuty and VictorOps.

**PagerDuty**

**Azure Integration:**

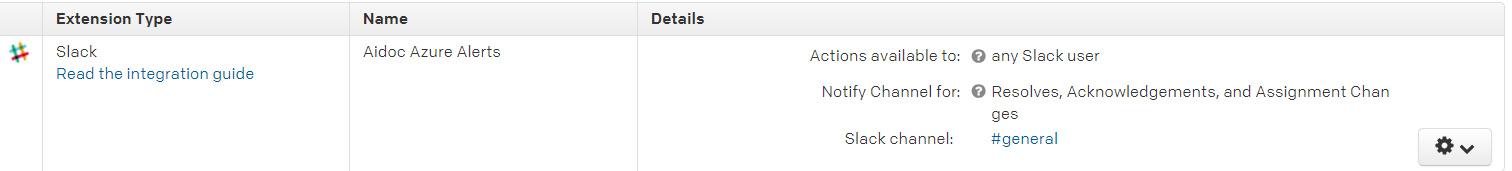
****

We first create a “service” that is attached to Azure.

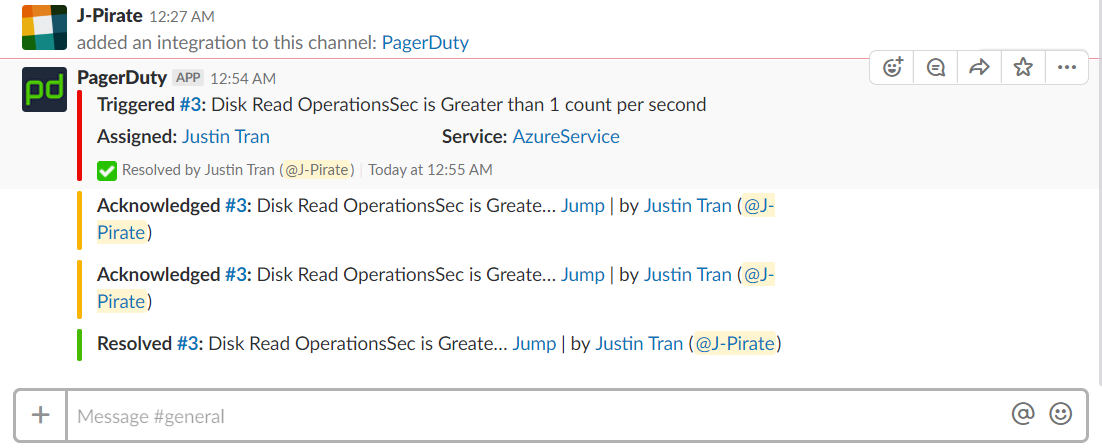
****

Webhook URL for Azure to direct its webhooks to.

**Slack Integration:**

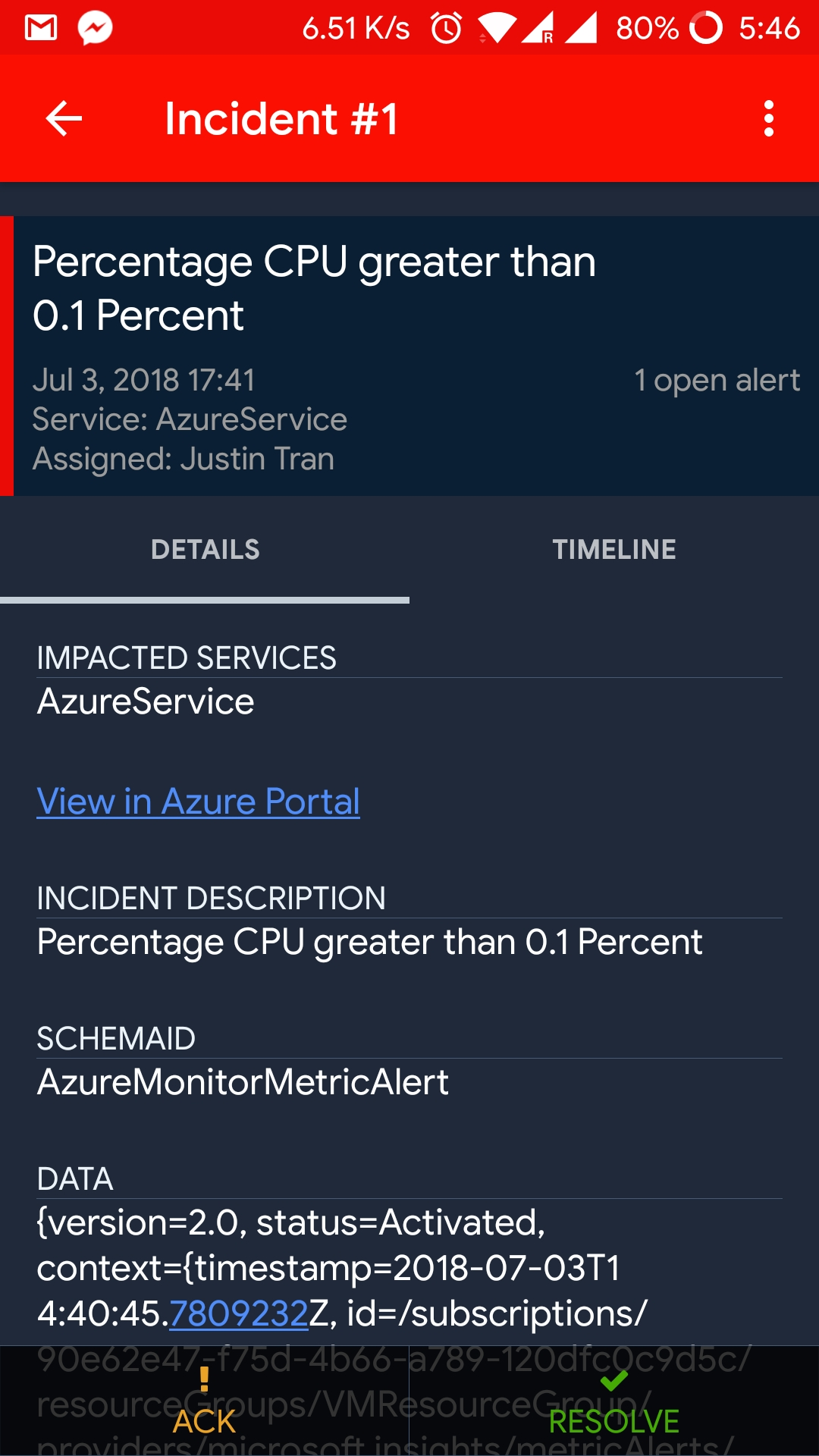
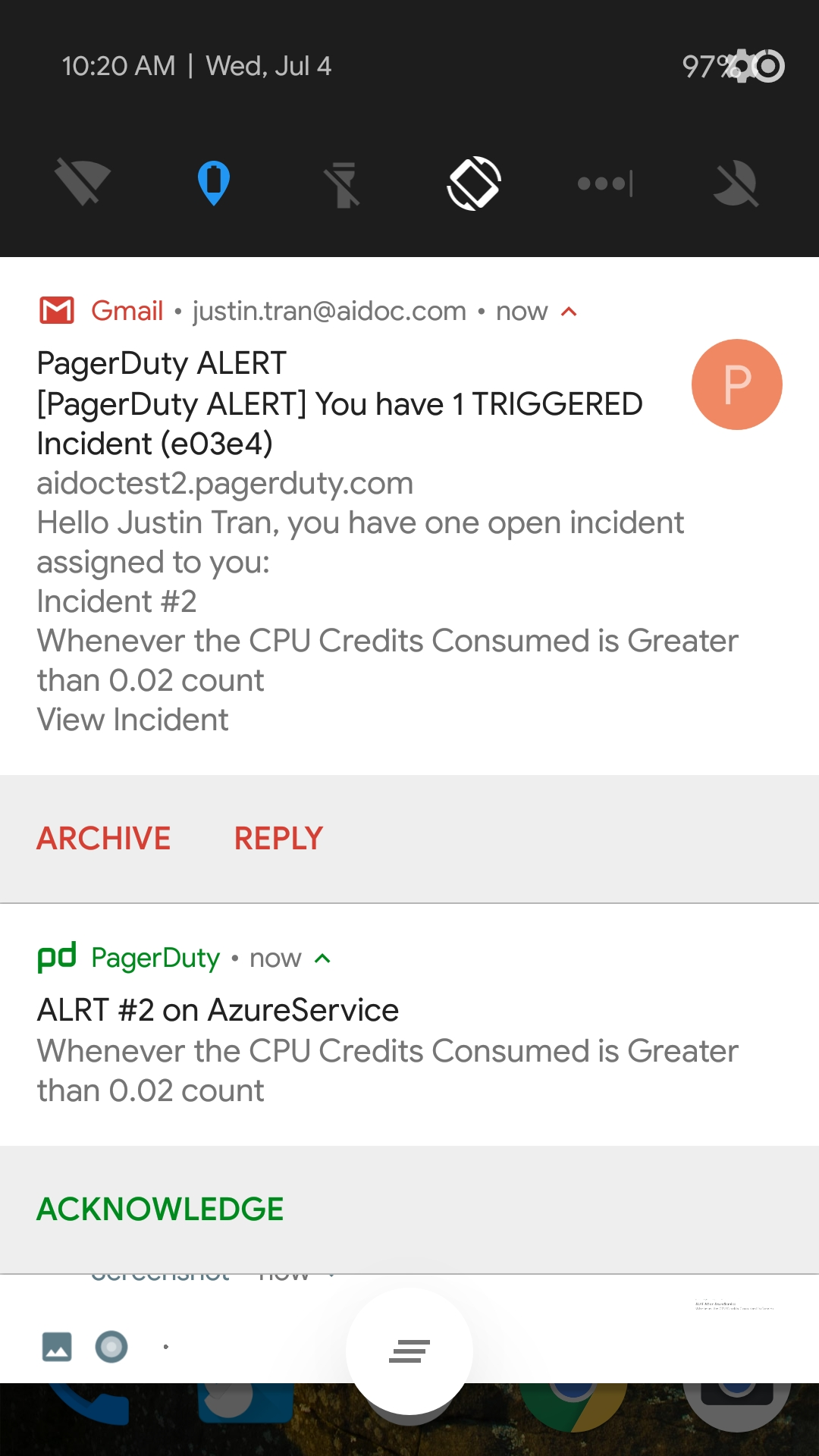
****

Notifies users in event of incidents, resolutions and acknowledgement of incidents, and changes to incident assignment.

****

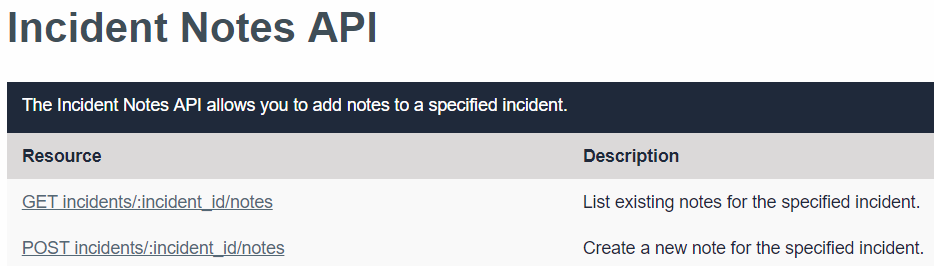
The PagerDuty Slack integration pings the entire channel or specific users and allows them to immediately acknowledge, resolve, or read more details of the incident.

**Mobile App:**

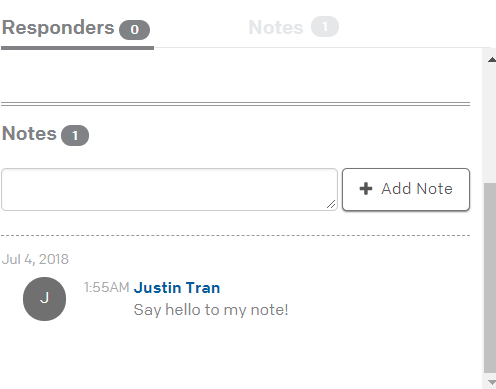
****

The mobile app immediately gets notified (faster than email) of the incident where the user can acknowledge or resolve the incident immediately.

**Metrics and Graphs:**



There is limited functionality for adding notes or creating a thread for an incident compared to VictorOps. PagerDuty only allows the capability for adding and viewing text notes to an incident as you can see even in the API.



Above is the web interface for adding incident notes which also only allows text.

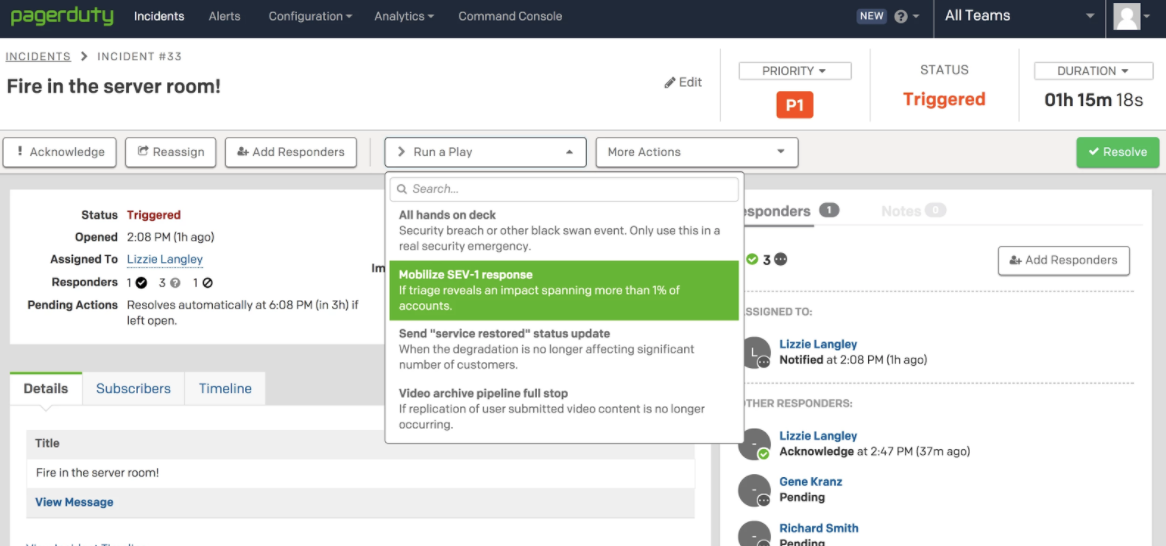
**Response Plays:**

This enables you to take a complex activity, like assembling a response team of multiple on-calls and an incident commander, and make it available to anyone that needs to use it. Grouping up these actions into a reusable play has several benefits:

* Turn documented processes into executable objects.
* Reduce the opportunity for errors and mistakes while following incident response processes.
* Drastically cut down on the time taken to execute incident response processes.

What can a response play actually do?

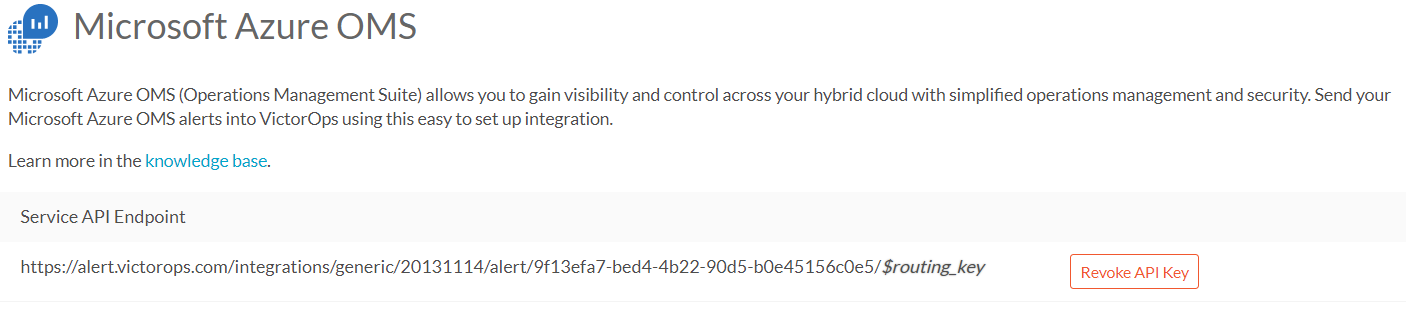
* Mobilize a Response (Notify responders other than the assignee to help resolve the incident.)
* Engage Stakeholders (Subscribe people who are not directly involved with resolving the incident to receive status updates.)
* Publish Status Updates (Send a prepared status update to all subscribers)



Responses can be triggered manually or automatically

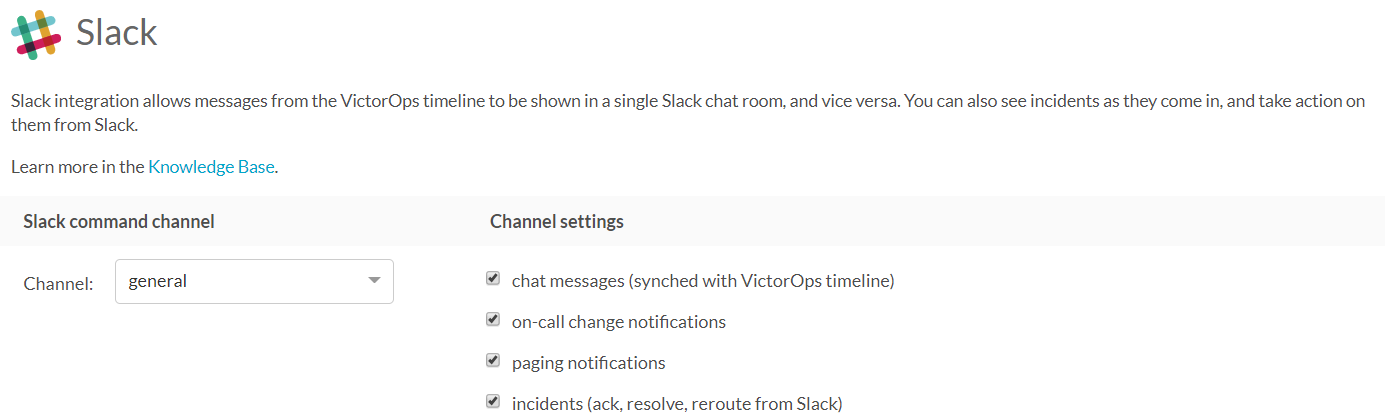
**VictorOps**

**Azure Integration (does not work):**

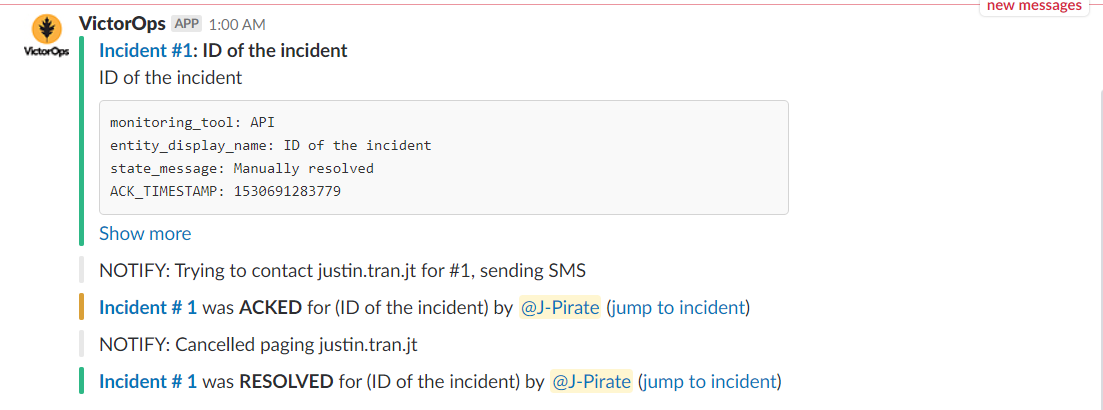
****

It turns out that VictorOps does NOT have integration with the current version of Microsoft Azure. A few months ago, Azure OMS was retired and integrated into the standard Microsoft Azure platform. **VictorOps has not updated their integrations with the Azure platform and as a result, no alerts in Azure can be sent to VictorOps. It must be done manually.**

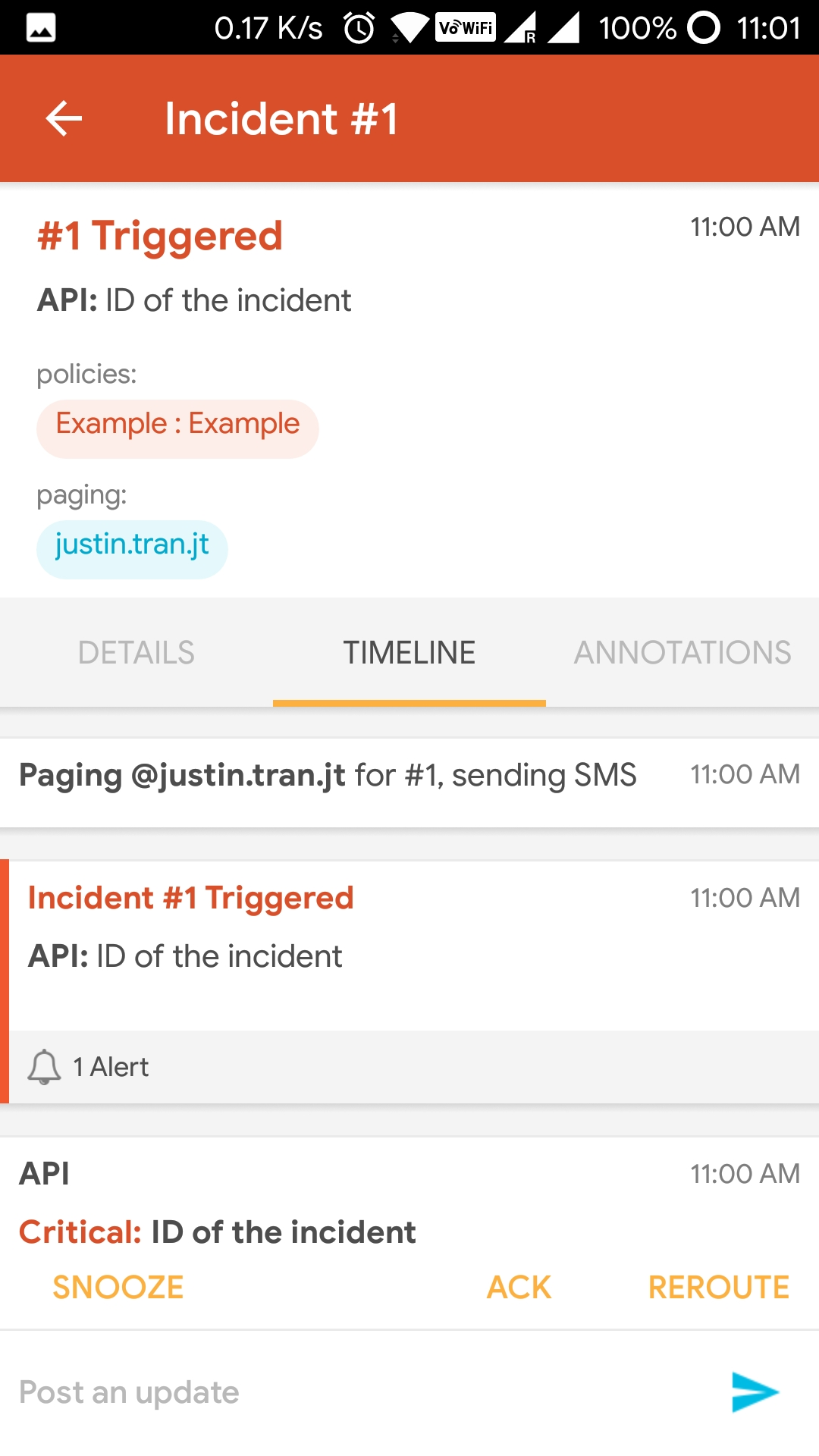
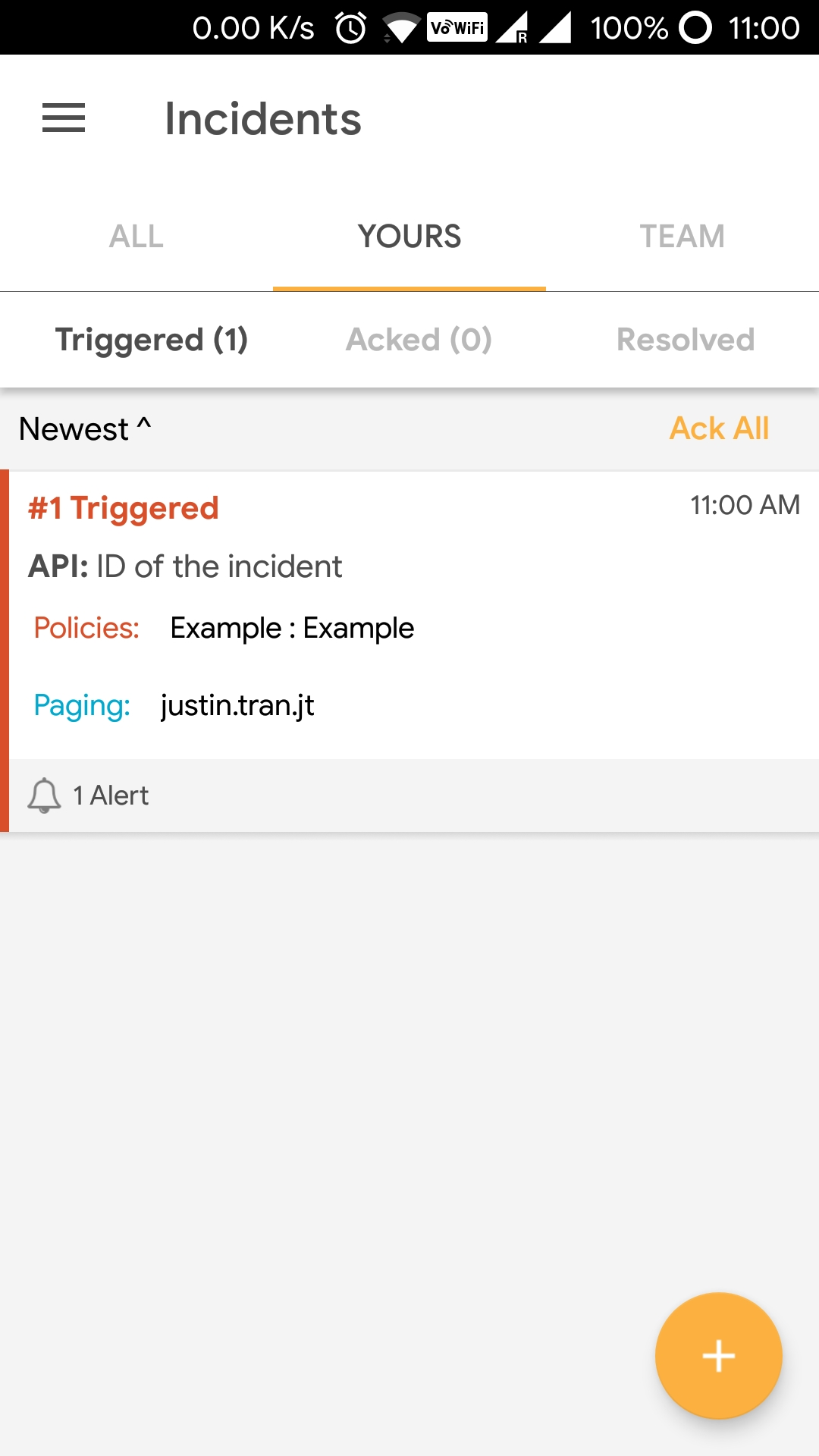
**Slack Integration:**

****

Slack integrations allows for direct incident management and chat messages from the VictorOps timeline.



**Mobile App:**

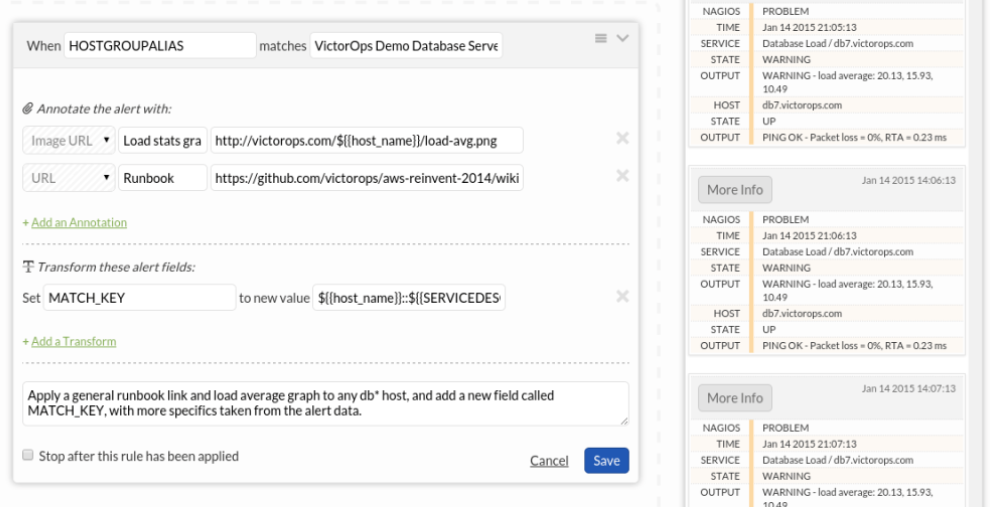
****

Here is an example of what the mobile app looks like. Incidents are presented on the front page with a description of the error. Clicking on the incident then allows us to view more details and acknowledge and resolve incidents.

**Metrics and Graphs (Transmogrifier):**

****

VictorOps allows for the inclusion of text, links, images, or Runbooks. It is, essentially, a rules engine that allows you to set certain conditions, and trigger custom actions when those conditions are met to make the incidents team members receive more detailed.

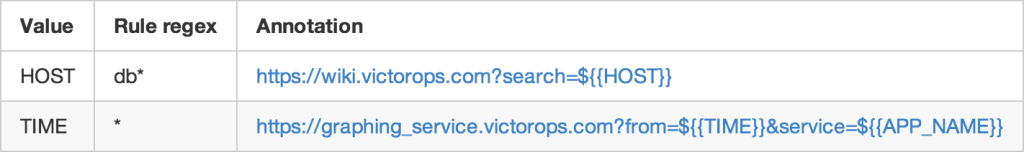


Here we see a way to query matching incidents and attach appropriate URLs and images that can provide information to users that view the incident. It can also transform fields by changing existing ones or adding new fields.

**Runbooks:**

VictorOps Runbooks are quite different in its goals compared to PagerDuty’s Response Plays. Runbooks aim to triage a problem and describe the steps towards how to fix the problem.

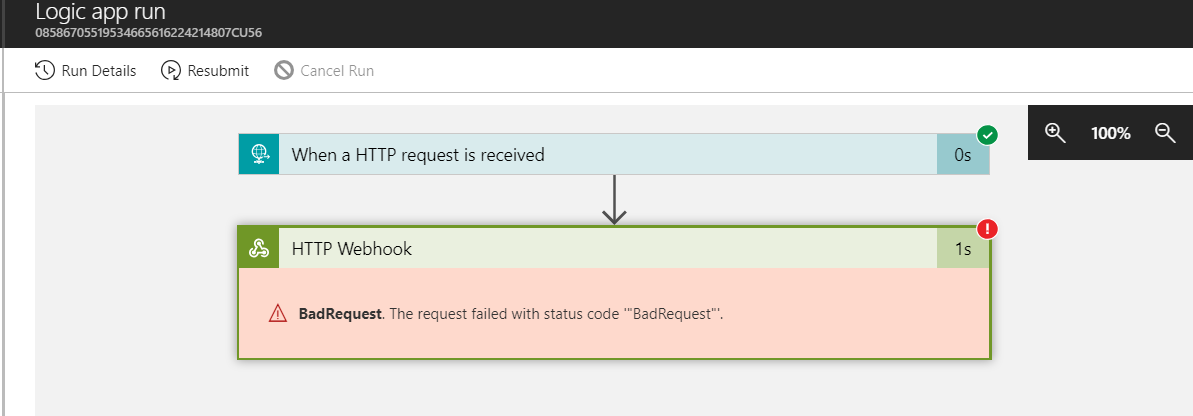
VictorOps Runbooks are mainly linked by the transmogrifier to show informative data that is dynamically generated by Github Wiki pages. For example:



This is an example of an annotation found in the Github Wiki page that creates the runbook and provides information. As you can see in the second row, a graph displaying important metrics is dynamically generated by a service such as **Graphite** (similar to the graph seen on the previous page) and is linked to the appropriate incident with this Runbook annotation.

**Attempting to Resolve the VictorOps Webhook with Logic Apps:**

Attempts were made to create a Logic App to send a JSON package to the VictorOps webhook. The Logic App webhooks were unsuccessful.



Multiple webhooks for VictorOps were attempted and **each resulted in a *BadRequest***.

**Attempted VictorOps webhooks:**

1. REST webhook: <https://alert.victorops.com/integrations/generic/20131114/alert/cec1bd2c-db64-43b4-ba1a-9cca04a01146/devops>
2. Microsoft Azure OMS: <https://alert.victorops.com/integrations/generic/20131114/alert/9f13efa7-bed4-4b22-90d5-b0e45156c0e5/devops>
3. Delivery Insights (accepts messages from custom sources via an HTTPS POST request in JSON format): <https://alert.victorops.com/integrations/dev/generic/20170920/05919b37-0abc-472a-bf04-955428724c20>

**Final Summary**

PagerDuty’s strong integration with Azure and alerts triggered by various VM conditions set by the user allows for seamless integration with the existing platform. Because Aidoc’s infrastructure depends so heavily on running virtual machines on Azure, it is obviously essential for whichever Incident Management Platform we choose to easily work with Azure alerts.

Despite this strong integration, PagerDuty is lacking compared to VictorOps when it comes to dynamic responses to incidents. The ability to attach images to incidents is not essential but it would be nice to have in PagerDuty. Unfortunately it only allows for text notes at the moment. Response Plays are also lacking as they simply notify the correct team members for specified incidents.

On the other hand, VictorOps and its Transmogrifier and Runbooks feature are very fleshed out and can provide dynamic content to each incident if it is relevant. Being able to see metrics on an incident page immediately is very helpful and definitely saves time.

Note that both mobile apps and Slack integrations are adequate at notifying users of alerts and allowing them to respond right from the interface by acknowledging and resolving incidents.

**However,** the dealbreaker with VictorOps is its lack of integration with Azure. It is still attached to the retired Azure OMS platform which has been discontinued and integrated into Azure. Alerts cannot be sent from Azure to VictorOps based on my extensive testing with webhooks and test alerts. Because these alerts are the main reason we want to use an incident management platform, **PagerDuty should be the platform of choice.** It works perfectly with Azure and while it may not have all the features of adding detailed information to incidents, the text notes are adequate and you can still view Azure alerts by clicking on a link in the incident.

**References**

* PagerDuty
  1. <https://www.pagerduty.com/docs/guides/azure-integration-guide/>
  2. <https://www.pagerduty.com/docs/guides/slack-integration-guide/>
  3. <https://support.pagerduty.com/docs/response-automation>
* VictorOps
  1. <https://help.victorops.com/knowledge-base/microsoft-azure-oms-integration-guide-victorops/>
  2. <https://victorops.com/slack/>
  3. <https://help.victorops.com/knowledge-base/transmogrifier/>
  4. <https://victorops.com/blog/victorops-runbooks-beta-goodness>