Developers Guide

Monitor Lizard is first and foremost a Django project, and as such to contribute to Monitor Lizard you must first learn how to develop for Django projects. Once you do this most of Monitor Lizard’s components are simple to understand.

To develop for Monitor Lizard, the first thing you probably want is a test environment. To set one up you can follow the installation guide, and read the notes for setting up a minimal deployment. Once you have this, remember to restart any components if you make changes to the code.

We will now walk through each component and note anything that is not considered standard Django.

## Database

The database is managed by standard Django models and migrations. We recommend learning the purpose of each model and their relationships before changing anything.

|  |  |  |
| --- | --- | --- |
| Term | Definition | Details |
| Metric | A metric is something that can be measured on a host. This can be anything from the percentage of disk used, to the exact byte count of disk used, to a list of disks in the system. | Each Metric should have a corresponding Probe that measures it in the host\_plugins folder.  The populate\_probes command (called by the setup command) automatically manages the Metrics to maintain the 1-1 relationship between Metrics and Probes. |
| Host | A computer that has registered via a host\_daemon. | The guid is what hosts use to identify themselves because anything else could change. We explicitly chose not to use ip address or host name, because saving the file on the host provides more consistent and predictable behavior.  The “get\_metric” method gets the latest value of a metric that was reported. |
| Host Tag | A collection of hosts. Hosts can be in multiple tags. The two primary purposes of tags are to make it easier to find hosts, and to allow users to assign the same alert rule to many hosts in a way that makes sense. |  |
| Report | A report is a measurement of one metric, on one host, at a point in time. | In the application “Report” actually refers to two things. The “reports” sent from the host\_daemon to the message bus actually contain multiple “Reports” worth of information, then the report is deconstructed in the processing daemon into Django model Reports. |
| Team | A team is a collection of users that are all subscribed to the same alerts. |  |
| User | A user that can receive alerts. | Monitor Lizard users are not Django users, and cannot login. Django users cannot receive notifications. An empty notification\_email or notification\_phone means the user has opted not to receive that kind of notification. |
| AlertRule | A rule that designates when alerts should be thrown.  The host tag is the collection of hosts that can trigger the alert rule.  The team is the collection of users that will be notified when the alert rule is triggered.  The operator and threshold together determine when a host triggers the alert rule. For example, the operator “min”, the threshold “80”, and the metric “cpu\_usage” means the alert will be triggered the first time any host in the host\_tag reports a cpu\_usage below 80. | The operator/threshold model is meant to be extensible. For example, if a process\_list metric was added, a developer could write an operator called “does not include”, with a threshold of “nginx”. The alert would be triggered any time the process list is reported not including nginx (meaning the process stopped running on the system). Another example operator is “max 7 day average”, where the threshold only trips if the 7 day average is above a certain threshold. |
| Alert | An alert that was sent out to users because an AlertRule was triggered. The alert should contain all of the information necessary for a user to investigate and diagnose the reason for the alert. The user can add notes to the alert, and acknowledge the alert. An acknowledgement means the alert has been investigated. | When creating Alerts, use the create classmethod. Because alerts can stay unacknowledged indefinitely, they must keep the information necessary to resolve themselves even if other models in the database are modified or deleted. The create classmethod copies this data over automatically.  If an alert has not been acknowledged, the alert daemon will not send more alerts. An acknowledgment of an alert is essentially an acknowledgment by a user that this alert is resolved, and any future triggering of the associated alert rule should be considered a different alert. |

## Web Dashboard

The web dashboard is a mostly permissionless tool for viewing data in the system. It uses standard Django, Bootstrap, ChartJS, and AlpineJS. The urls are split between the files in the urls folder, and the controllers are split between the files in the controllers folder.

## Message Bus

We use the pika library to communicate over AMQP to our message bus. The configuration and library-wrapping are in message\_queue.py. The send method sends a string, and the receive method accepts a callback which it passes to the on\_message\_callback parameter of [this pika method](https://pika.readthedocs.io/en/stable/modules/channel.html#pika.channel.Channel.basic_consume).

## Processing Daemon

The processing daemon uses message\_queues receive method to receive reports over the message bus. It decodes the json into a HostReport, which it then turns into django model reports. Every 100 reports the processing daemon saves the model reports to the database.

## Host Daemon

The host daemon makes sure the host it's running on is registered, measures probes, and sends the data to the message queue at a regular interval.

First it checks if the host has a guid saved in a file. If it doesn’t, it tries to contact the web dashboards api to get a guid. If it is able to get a guid, then it knows it is registered in the database as a host, and can begin sending reports.

Before the host daemon begins measuring probes, it must first collect them. It collects probes by looking for classes named ProbePlugin in files in the host\_plugins folder. Each ProbePlugin extends HostPlugin and knows how to measure every probe in itself using measure(). measure() currently accepts a dictionary of the last time the metrics were polled, and the host specific configuration for polling intervals, but always measures probes ignoring this information.

## Alert Daemon

The alert daemon continuously reads all alert rules in the database, and checks to see if any host triggered the alert rule. An alert rule will not trigger for a particular host if it has already been triggered for that host and not been acknowledged.