Mender

Automated configuration discovery

Mender By OPSDIS

Mender overview

IT infrastructure has become more dynamic. With the increasing use of cloud services, container orchestration, dynamic scaling of services etc, the landscape of our IT environment is changing at an accelerating pace. Many new architectures support this type of infrastructure by using different kinds of discovery patterns. For infrastructure and application entities, discovering changes can be used to provide different services like scaling, deployments, change load balancers, health checking etc.

For service discovery to work, however there is a need for sources of information. Changes in the source system trigger events that the discovery service turns into action(s).

When it comes to monitoring, systems discovery must be the correct way to configure what to monitor. In modern systems like Prometheus, service discovery is an integrated component. E.g. if the discovery detects that a new host has been added to a DNS server it will automatically start to scrape metrics from the new host without any need for reconfiguration or restarting the monitoring service. So in Prometheus it is separated into configuration (i.e. what to do), and discovery (what to do it for).

For traditional monitoring systems this is typically not the case. Typical configuration is a manual process, which causes a number of negative effects:

- Long latency before new or changed hosts and services are added to the monitoring system. Often there are different organisations that are handling the setup and deployment of applications/services/infrastructure and the monitoring system itself.
- The same latency or lack of feedback is also seen when hosts or services are removed or decommissioned, which can lead to false alarms.
- In a manual process it's often the case that different people will do the monitoring configuration in different ways, leading to a wide spectrum of standards and policies, even if they are documented.
- Manual processes consume a lot of time while keeping up with changes in the environment, which leads to a high labor cost of professional engineers.

Mender overview

The overall result is that we have a monitoring system with low data quality that does not reflect the real IT environment, which in the worst case could lead to critical incidents are not detected or not possible to resolve quickly.

Mender is our solution to this problem. It is targeted to enable automated configuration discovery for the ITRS OP5 Monitor monitoring system. OP5 Monitor has its roots in the classical Nagios data model, but has evolved to a powerful monitoring system including powerful GUI, high-availability, API's for configuration and operational data. But configuration automation has been left up to the user to implement. Since OP5 Monitor typically targeted large organisations with substantial IT arsenals, configuration automation is a key aspect to provide effective and up-to-date monitoring.

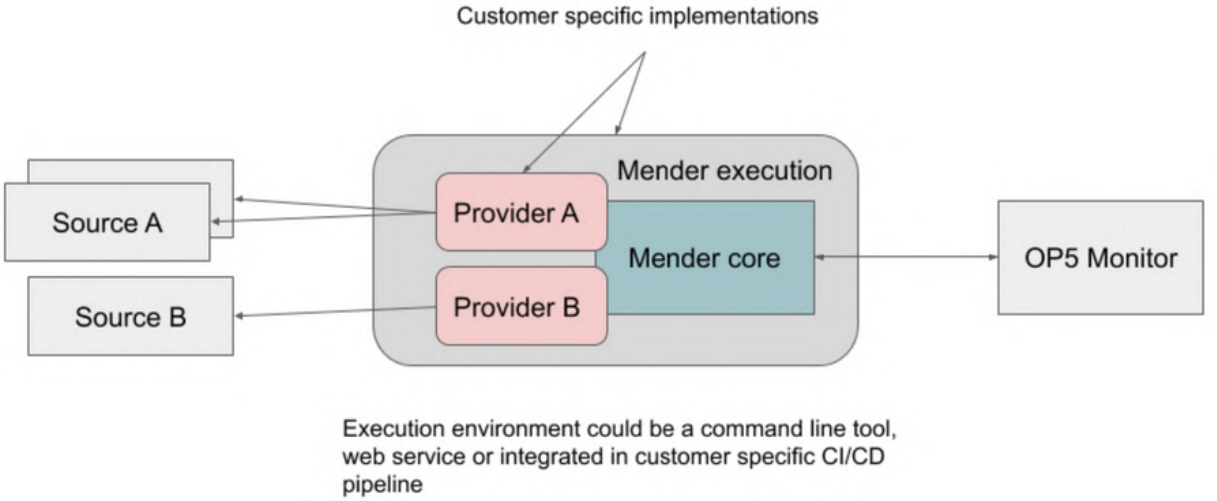
So what does automated configuration discovery mean in the context of OP5 Monitor?

The idea is to connect different source systems that contain information that can be used to create configuration of hosts and services in Monitor. There can be multiple and different kinds of source systems, network management system e.g. Cisco Prime, a traditional cmdb system like ServiceNow, a configuration system like Puppet, or IPAM system like Infoblox etc. It is important that these systems contain the "truth" of the IT environment that is deployed related to servers, switches, routes, load balancers, databases, messages buses, applications etc.

This is what Mender does, integrating all these systems with Monitor by managing any life cycle changes in the source system and comparing it with the configuration in Monitor for the equivalent objects. If different from the source, Mender will update Monitor with only what has changed using the public Monitor API's. The comparison processing can be run on schedule, like every hour, or on demand depending if a state change can be detected in the source system.

Mender overview

Mender is a Python module and can run on every server platform that supports version 3 of Python. The integration with the source system is done with what is called providers, which are Python modules. Which provider modules to run is defined in a Mender configuration file. This enables full customization of how data is extracted from the source system, while Mender will just manage all the logic and processing in regards to interacting with Monitor.



Architecture overview

With Mender, our customers will always have a monitoring system up to date with the environment it is responsible to monitor. Manual configuration will be minimal as a complete automated process for configuration is implemented.

Mender architecture

The core of the Mender architecture is a generic object model of Hosts, Services and Hostgroups. This model is populated in two steps, by first extracting the information from Monitor and then extracting the information from the source system by a provider. When the two model stacks are populated, Mender conducts a comparison between them to calculate the difference between the two. This difference can resolve in that an object needs to be created, updated or deleted. Only the objects that are different are managed using Monitor's configuration API. This keeps the number of API calls to a minimum. When all the configuration calls have been executed, Mender calls the Monitor exports API to deploy the configuration and restart Monitor. This is the same process as if the changes had been done from the Monitor Web GUI manually.

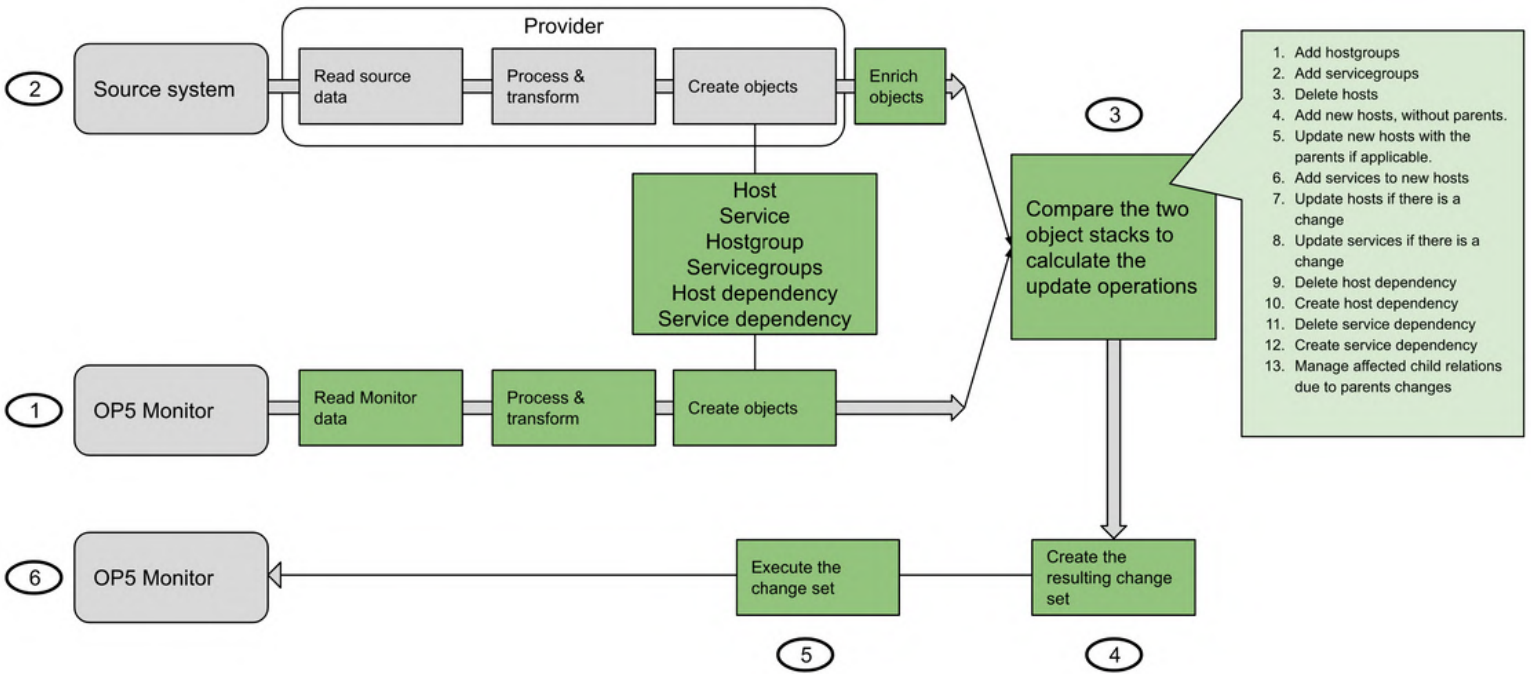
Mender architecture

Since a Mender setup can have multiple providers, some customers have up to +10 different source systems, Mender needs to know for all the hosts in Monitor from which provider they originated from.

This is done by using a hostgroup that is unique for each provider that all hosts configured from the provider will belong to. This is called a provider "marker" hostgroup and is used when selecting hosts and their services from Monitor for comparison with a specific provider.

The processing flow done in 6 steps:

1. Read all host and service configuration for the provider marker hostgroup and create the Mender object stack
2. Read the source system data related to hosts and service and create the Mender object stack.
3. Compare the two stacks Create a result list of differences that need to be executed against Monitor to get a consistent state.
4. Execute the API calls for the above list.
5. Save the new configuration and restart Monitor.



Mender architecture

The processing of the above steps is done in two phases. In the first phase, Mender resolves all the needed configuration actions according to the steps above and creates a list of API calls that must be executed to achieve the new state of the provider. In the second phase, each API call is done according to the list. The reason for this two phase approach is that each step in each phase is communicated to a callback class that can be customized. This callback can be used to subscribe to the events occurring and actions can be taken, like updating some third party system, creating a specific audit log or even aborting future processing.

With Mender we can create Monitor configuration with thousands of hosts and tens of thousands of services in a fully automated and repeatable way.

Operational features

Mender has a number of configuration options that can be set in its configuration file. Some examples are:

- Do not process hosts and service currently in scheduled downtime
- Strict handling of hostgroups and contactgroups. E.g. if a host is only allowed to belong to hostgroups defined by the provider or not.
- Services on a hostgroup will show up as a "normal" service on the host that belongs to the hostgroup. These services should normally not be part of the comparison process.

In Monitor, the processing of a host or service can also be controlled by using custom variables. Currently Mender supports the custom variable _EXCLUDE. If set to True the host or service will be excluded from any future processing until unset. This should be used with care but if manual changes have to be done this will ensure that the object is not overwritten during the next Mender execution.

Mender providers

Providers are the Python modules that must be developed to retrieve data from the source system. A provider must follow a simple Python interface to be executed from Mender. Here are some examples of providers that have been developed for customers or by customers.

NETWORK MANAGEMENT SYSTEMS

- Cisco Prime
- Cisco ACI
- NetMRI
- Nedi
- FortiManage
- Fortigate
- Fortiswitch
- Aruba Airwave
- Infoblox
- Biglp (F5)

CONTENT MANAGEMENT

- GIT
- Confluence

CLASSIC CMDB

- Kace

INFRASTRUCTURE PROVISIONING AND DEPLOYMENT TOOLS

- Puppet
- AMQ broker

MONITORING TOOLS

- OP5 Monitor
- Prometheus

ORCHESTRATION PLATFORMS

- Rancher
- Openshift

A number of customer specific inventory systems are also a part of this list of providers.

A provider has the following responsibilities:

1. Read adequate data from the source system. This is typical over API exposed by the source system
2. Transform the data into Host, Hostgroup and Service objects.

After these steps, Mender will do all the heavy lifting to update Monitor to the correct state.

Mender license

Mender is licensed under the GPL version 3 license. Subscribing customers have access to new versions on Opsdis private pip repository and support.

Case studys

Our customers are using the Mender solution with a variety of configuration. Read about some of them below

- VGR IT & Opsdis – An automation journey that increased quality and efficiency
- Using monitoring to stay competitive in the 24/7 gambling industry
- Combining Promethes and Nagios

Contact

Hampus Markkrona
Sales and Marketing

Hampus.markkrona@opsdis.com
+46 73-514 78 61

Johannes Dagemark
CEO and co-founder

Johannes@opsdis.com
+46 73-370 90 24