### Commands

To update the policies and enforce them
```
rucker agent run -u
```
To see detailed output
```
rucker agent run -i
```
To trigger an inventory
```
rucker agent inventory
```

### Units

In all relevant fields (group, search, etc.), available units are B, KB, MB, GB, TB, PB, ...
When unit is omitted, is is considered as a value in bytes.
Use spaces or underscores to format the number (1,024 or 2,048).

### Advanced search syntax

You can use a query language to specify your search in the search bar:

<table>
<thead>
<tr>
<th>Query</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>is:node in:ips 192.168.1.2</td>
<td>Return nodes with ip 192.168.1.2</td>
</tr>
<tr>
<td>is:node</td>
<td>groups</td>
</tr>
<tr>
<td>in:property_name</td>
<td>Return the property you are searching in</td>
</tr>
<tr>
<td>is:= in:name</td>
<td>id</td>
</tr>
<tr>
<td>is:node</td>
<td>hostname</td>
</tr>
<tr>
<td>rami:policy_server_id</td>
<td>properties</td>
</tr>
<tr>
<td>is:group</td>
<td>in:dynamic</td>
</tr>
<tr>
<td>is:directives in:direct_param_name</td>
<td>dir_param_value</td>
</tr>
<tr>
<td>is:parameters in:parameter_name</td>
<td>parameter_value</td>
</tr>
<tr>
<td>is:rules in:directives</td>
<td>groups</td>
</tr>
</tbody>
</table>

### Conditions

Conditions are context entities represented by a string, that can be either set or not set, depending on context. They allow using conditions to generic method usage. Variable name must match:
```
[a-zA-Z-0-9][a-zA-Z-0-9-_.]*
```

**Available conditions**

Group conditions: defined only if the node is in the given group (available in the group details)
```
group_group_name
group_group_name
```

System conditions: various system information defined by default
```
centos_7_1, ubuntu_18_04
```

Result conditions: defined by the execution of another generic method (available at the bottom of the generic method call configuration)
```
generic_method_name_parameter_value_kept
generic_method_name_parameter_value_repaired
generic_method_name_parameter_value_error
```

Conditions manually defined in the agent call:
```
rucker agent run -D my_condition
```

**Condition expressions**

```
(condition_expression) Or (condition.Other)
```

**Paths**

**On the nodes**
Policy server configuration file
```
/opt/rudder/etc/policy_server.dat
```

**On the server**
Directory containing all the configuration policies in a git repository
```
/var/rudder/configuration-repository
```
Directory shared with nodes from the server
```
/var/rudder/configuration-repository/shared-files
```
Directory containing the configuration events (change and errors)
```
/var/log/rudder/compliance/non-compliant-reports
```
Webapp log file
```
/var/log/rudder/webapp/YYYY_MM_DD.stderrout.log
```

### Variables

Variables in directives parameters are evaluated at generation on the server, exceptions are tagged with execution.

Variables in technique editor are evaluated at execution on the nodes.

Node properties can be overridden at execution on the nodes using files containing a "properties" object placed in:
```
/var/rudder/local/properties.d/*.{json}
```

**Mustache Templating**

**Conditions**
```
{{#classes.conditions}}
condition is defined
{{/classes.conditions}}
```
**Variables**
```
{{vars.node.properties.variable_name}}
```

**Jinja2 Templating**

**Conditions**
```
{% if classes.condition is defined %}
condition is defined
{% endif %}
{% if not classes.condition is defined %}
condition is not defined
{% endif %}
```
**Variables**
```
{{vars.prefix.dict_name[variable_prefix.string_name]}}
```

**Useful links**
- [www.rudder.io](http://www.rudder.io)
- [chat.rudder.io](http://chat.rudder.io)
- [docs.rudder.io](http://docs.rudder.io)
- [issues.rudder.io](http://issues.rudder.io)