Upgrading to Scalyr Agent 2

This document takes you through the process of upgrading from the original Scalyr Agent. If you are trying to install Scalyr Agent 2 on a new server, then please follow the instructions at https://www.scalyr.com/help/install-agent.

Upgrade Instructions

Scalyr Agent 2 is an entirely new package, not an update of the original Scalyr Agent. The following instructions walk you through installing the new package, copying your configuration from the old agent, halting the old agent, and launching the new agent. Don't be daunted by the length of these instructions; you'll be able to skip many of the sections, depending on which agent features you're using. A typical upgrade should take 10 to 15 minutes, plus a few minutes per additional server.

1. Package installation

Currently, the Scalyr Agent 2 is available through our yum or apt repositories. If you'd like a tarball version, e-mail us at support@scalyr.com.

For systems using yum or apt, the following script will update your Scalyr repository and fetch the new scalyr-agent-2 package:

```
wget https://www.scalyr.com/scalyr-repo/stable/latest/install-scalyr-agent-2.sh
sudo bash ./install-scalyr-agent-2.sh
```

Upon success, you should see a command that instructs you how to start the agent. Do not perform that command yet; first, you need to migrate your agent configuration file.

2. Configuration

Your existing configuration is in `/etc/scalyrAgent/agentConfig.json`, or `~/scalyrAgent/config/agentConfig.json` if you installed from a tarball. Configuration for the new agent is in `/etc/scalyr-agent-2/agent.json`. Both files are JSON, but the details differ. You'll have to manually copy several pieces of information to the new agent's configuration file.

Copy the API key

In your existing agentConfig.json file, you should see a line such as:

```
"scalyr_client.write_logs_key": “abc123abc123abc123”,
```

In the new agent's agent.json file, there should be an `api_key` field, with value `REPLACE_THIS`. Fill in your API key:

```
api_key: “abc123abc123abc123”,
```

Copy server attributes

You may have specified some fields to be attached to all log messages sent from this host. These are commonly used to override the server's hostname, or attach other information such as the region in which the server is located.

In the old configuration file, these attributes are in the `scalyr_client.global_attributes` section. It will look something like this:

```
"scalyr_client.global_attributes": {
   server_group: "production",
   region:       "us-east-1"
   serverHost:   “big-server-1”,
},
```

In the new file, the section is named `server_attributes`:

```
server_attributes: {
   server_group: "production",
   region:       "us-east-1"
   serverHost:   “big-server-1”,
},
```

Copy log configuration

Next, copy the configuration for the log files your agent is sending to Scalyr. This will take a little more work, because the new agent uses a different (simpler) syntax.

The relevant parts of your agentConfig.json file will be in the `log_copier.config` section. Specifically, you are looking for the `directories` field in that section. It will look something like this:

```
logcopier_config: {
  directories: [
    {
      path:"/var/log/tomcat6",
      match_expressions: ["access_log\\.txt"],
      default_attributes: {source:"tomcatAccess"}
    }, {
      path: "/home/ec2-user/sample-log.txt",
      file_attributes: {source: "fakeTest"}
    }
  ]
},
```

This example shows the different ways that you could have used to express your log configuration. In particular, in the second entry, the full path for the log file is in the `path` field as `/home/ec2-user/sample-log.txt`. This is different than the first entry, where the path only specifies the path for the directory, while `match_expressions` contains a list of regular expressions that should match the file name within that directory. Additionally, `default_attributes` and `file_attributes` are different field names for the same configuration option, the fields that are included in all log entries from those files.

Your log configuration may also contain redaction and sampling rules. If so, please see the section below that describes how to copy those options.

The chief differences between the old and new log configuration format are:

- The nested `logcopier_config` and `directories` fields have been collapsed to a single `logs` field.
- Log file paths are no longer split into separate `path` and `match_expressions` fields, and are no longer grouped by directory. Each path (or glob) uses a separate entry.
- You can only use glob patterns in file paths. You cannot use regular expressions.
- The `default_attributes` / `file_attributes` field has been renamed to simply `attributes`.
- The `source` attribute has been changed to `parser`, since the primary use of this field in Scalyr is to indicate which parser should be used to parse the log.

Edit `/etc/scalyr-agent-2/agent.json` to list your log files using the new syntax. The example from above would become:

```
logs: [
  {
    path: "/var/log/tomcat6/access_log.txt",
    attributes: {parser: "tomcatAccess"}
  }, {
    path: "/home/ec2-user/sample-log.txt",
    attributes: {parser: "fakeTest"}
  }
],
```

Copy redaction and sampling rules

If you have used some of the advanced features of the agent, some of your logs may be using sampling or redaction rules to limit what data is sent to Scalyr. These rules are specified for each directory or log they apply to. You'll need to copy each redaction or sampling rule to the new configuration.

The example below shows how a single entry may look in your existing `agentConfig.json`:

```
logcopier_config: {
  directories: [
    {
      path:"/home/ec2-user/sample-log.txt",
      file_attributes: {source:"tomcatAccess"}
      sample: [ {pattern: "INFO", fraction: 0.1} ],
      replace: [
           "password=[^& ]*",
           {pattern:"(secret_[^=])=[^& ]*", replacement: "$1=hidden"},
      ]
    },
  ]
},
```

This example shows both sampling and redaction (`replace`) rules. It shows two different formats for `replace`. The first is a single string that indicates that any match to the regular expression should be deleted. The second indicates that each match for the pattern should be replaced by the specified replacement.

In the new agent:

- `sample` becomes `sampling_rules`
- `replace` becomes `redaction_rules`
- `pattern` becomes `match_expression`
- `fraction` becomes `sampling_rate`
- The shortcut format for redaction rules is no longer supported; you must say `{match_expression: "pattern"}` rather than simply `"pattern"`.

The above example becomes:

```
logs: [
  {
    path: "/home/ec2-user/sample-log.txt",
    file_attributes: {source:"tomcatAccess"}
    sampling_rules: [ { match_expression: "INFO", sampling_rate: 0.1} ],
    redaction_rules: [
      { match_expression: "password=[^& ]*"}
      { match_expression: "(secret_[^=])=[^& ]*",
        replacement: "$1=hidden"},
    ]
  }
],
```

Copy per-process metrics

If you had instructed the agent to report any per-process metrics, you must copy these rules to the new configuration file. Look for an `appmonitor_config` section in `agentConfig.json`:

```
appmonitor_config: {
  applications: [
    {
      id: "tomcat",
      commandline_pattern: "java.*tomcat6"
    }, {
      id: "scalyr-agent",
      commandline_pattern: "java.*scalyrRelay.jar"
    },
  ]
},
```

This feature is now supported by a monitor plugin called “linux_process_metrics”. For each process you wish to monitor, add an entry for every process you with to watch in the “monitors” section of your new configuration:

```
  monitors: [
    {
      module: "scalyr_agent.builtin_monitors.linux_process_metrics",
      id: "tomcat",
      commandline: "java.*tomcat6"
     }
     // We do not list "scalyr-agent": this is now monitored by default.
  },
```

Enabling metric collection using Graphite

Similar to the original agent, Scalyr Agent 2 has the ability to accept metrics using the Graphite plaintext and pickle protocols. If configured, Scalyr Agent 2 will run a server as part of its process that will accept network connections on two ports (one for each protocol) and relay any received metrics to Scalyr. By default, this server will only accept connections from localhost, but can be configured to accept from any host.

You will need to determine if your system requires metric collection using Graphite. By default, the original agent always ran the Graphite server and accepted connections from any host so there may not be any mention of it in your original configuration file even if you required it. To be less surprising in what it does, Scalyr Agent 2 is now configured to not run the Graphite server by default.

To enable collecting metrics using Graphite, add the following to the the monitors section of your new configuration file:

```
monitors: [
  {
    module: "scalyr_agent.builtin_monitors.graphite_monitor",
  },
   ...
},
```

This will accept Graphite traffic on the default ports of 2003 (plaintext) and 2004 (pickle) from localhost. If you wish to modify any of these parameters, such as accepting connections from non-localhosts or only accepting one of the two protocols, use the example below as a guide to add the appropriate options to your configuration:

```
monitors: [
  {
    module: "scalyr_agent.builtin_monitors.graphite_monitor",
    only_accept_local: true,  // Set to false if accept connections from
                              // any host, not just localhost
    accept_plaintext: true,   // Set to false to turn off accepting
                              // connections on the plaintext port.
    accept_pickle: true,      // Set to false to turn off accepting
                              // connections on the pickle port.
    plaintext_port: 2003,     // Change to use a non-standard port for
                              // for plaintext connections.
    pickle_port: 2004,        // Change to use a non-standard port for
                              // for pickle connections.
  }
   ...
},
```

3. Stop the existing agent

Once your configuration has been finalized, you are ready to start using Scalyr Agent 2. You should first stop the existing agent.

If you installed from the apt or yum repository, the command is:

```
sudo scalyr-agent stop
```

If you installed from the tarball, the command is:

```
~/scalyrAgent/bin/scalyr-agent stop
```

4. Start the new agent

To start the new agent, use the following command:

```
sudo scalyr-agent-2 start
```

At startup, the agent will verify it can parse the configuration file and communicate with the Scalyr servers. If not, an appropriate error will be reported to standard error and the daemon will not be started.

For troubleshooting issues with starting Scalyr Agent 2, please see https://www.scalyr.com/help/scalyr-agent-2

5. Removing the old agent

Once Scalyr Agent 2 is running successfully and you no longer need the original agent, you should uninstall it. This will prevent the agent from accidentally being relaunched again, especially if your machine restarts and initd decides to launch it.

For yum-based systems, use:

```
sudo yum remove scalyr-agent
```

For apt-based systems, use:

```
sudo apt-get remove scalyr-agent
```

If you installed using the tarball method, there is no single uninstall command. You can do:

```
rm -r ~/scalyrAgent/
```

Things to know

There are a few differences you should be aware of between the old and new agents. Most of these are just small changes and should not impact how you use Scalyr or the agent. However, some users may have to modify their alerts or saved searches to take these into consideration.

Removed deprecated support for "host" and "filename" fields

In the early days of the agent, Scalyr added a `host` field to log messages, to indicate which server generated the message and a `filename` field to indicate which log file it came from. However, this conflicted with the `host` field in web access logs and `filename` from user generated parsers. The agent was changed to add `serverHost` and `logfile` fields, but the old `host` and `filename` fields were retained for backward compatibility.

Scalyr Agent 2 now only uses `serverHost` and `logfile`. If you have dashboards or alerting rules that relied on `host` or `filename` you will need to update them to use `serverHost` and `logfile`.

Log file names now include full path

The original agent recorded a `logfile` field for each log message, giving the name of the log file. This field did not contain a complete path, only the filename.

The new agent reports the full file path in `logfile`. If you have alerting rules or dashboards which rely on `logfile` to be something like "access.log" instead of "/var/log/tomcat6/access.log", you will need to update them.

Log location

The new agent uses new paths for the logs and data it generates. Logs are now saved in `/var/logs/scalyr-agent-2` instead of `/var/logs/scalyrAgent`, and internal agent state is now stored in `/var/lib/scalyr-agent-2` instead of `/var/lib/scalyrAgent`.

Additional logs sent to Scalyr

In your overview page at https://ww.scalyr.com/, you will notice several new logs files appearing for each of the servers running Scalyr Agent 2. They are:

/var/log/scalyr-agent-2/agent.log
This is a new log collected by Scalyr Agent 2 containing any error messages generated by the agent process or monitors it is running, along with some useful metrics recording the performance of the agent. This can be used to help diagnose any issues you have with the agent.

/var/log/scalyr-agent-2/linux-system-metrics.log
This contains system metric information such as the host CPU and RAM usage.

/var/log/scalyr-agent-2/process_metrics.log
This contains per-process metrics for the agent itself, and any other processes you have configured the agent to monitor.

Configuration can be split across multiple files

With Scalyr Agent 2, you may split your existing configuration file across multiple files. In addition to reading configuration from `/etc/scalyr-agent-2/agent.json`, Scalyr Agent 2 will read configuration from any file ending in `.json` in the `/etc/scalyr-agent-2/agent.d` directory. These files can only contain `logs`, `monitors`, or `server_attributes` sections.

This modular configuration allows you to create separate configuration files for the different applications you run and can ease integration with automation tools such as Chef.