**Developer Documentation**

**Overview:**

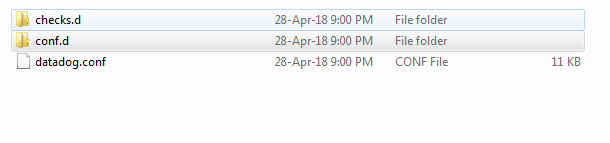
* **Datadog Agent:**

Datadog Agent is software that runs on hosts, which collects events and metrics for monitoring and performance of data. The agent has three main parts: collector, DogStatsD and forwarder.

* **The Collector**:- Collector runs checks on the current machine for whatever integrations you have and it captures system metrics such as memory and CPU.
* **DogStatsD**:- It is a statsd backend server you can send custom metrics to from an application.
* **The Forwarder**:- This retrieves data from both DogStatsD and the collector and then queues it up to be sent to Data-dog
* **Agent check:**
  + - Agent checks used to collect metrics from custom application or systems. It’s important to write integration for collecting all metrics available in a application.
    - In Agent Check interface, all the custom checks will inherit from /checks/ directory and it requires a check () method that takes one argument and an instance which is a dict having configuration of particular instance.
    - The check method is run once per instance defined in checks configuration.
* **Metrics:** 
  + - Metrics allows creating Query metrics from any time period and post metrics data which we can be graphed on Datadog Dashboards.
    - Metrics type visible on metrics summary page , developer should change the type if he want to submit metrics with a new type. There are three types of metrics. They are Gauge, Rate and Count.
    - There are many ways to send a metrics to Datadog.
    - With Datadog agent directly , by using StatsD or submit metrics directly to Datadog HTTP API .

**Configuration :**

* After all the installation of Datadog Agent . We need to go through /etc/dd-agent folder
* We have two Directories , one is “checks.d “and “conf.d”



* **Conf.d** contains **“.yaml “** files & **Checks.d** contains **“python files”.**
* The agent checks and some logic are written in python and they rely on yaml file configuration.
* We can write or modify existing checks to create and collect metrics
* The name of check file and config file must match.

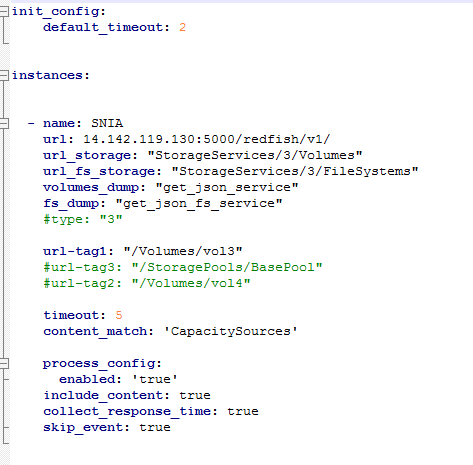
If we named checks as “**sample.py** “ then our config file must named as **sample.yaml.**

**Config files (YAML):**

* **Objective:** developer can configure server integration by adding url and sub tags. Each check has a YAML configuration file that is placed in conf.d directory. The file name should match the name of checks module.
* From below figure .yaml file contains init\_config and instances Section.

init\_config section allows to have configuration option available to run checks.

* Instances Section is list of instances that runs against checks.
* A developer can integrate Swordfish service by defining URL and some information regarding the service.
* Developer can add tag and sub-tags of resource which makes easy access to all the data in a particular server.



**Checks (python) files:**

* **Objective:** The main part of checks is making a request to url . Once request passes, write custom metrics for service url which is placed in .yaml file.
* All the custom checks inherit from AgentCheck and it will take check() to take arguments. Each check instance has its own instance of class.
* The check method run once per instance.
* Checks will make a request to URL and response time .
* Developer can send events , gauges and service checks in a check.
* From the below example its sending metrics in a check

Example: Self.gauge( )

The above method can take arguments like value, tag, hostname and metrics.

* All the custom checks are not able to import modules by default .
* Once we place the checks in checks.d folder we can test it by restarting the agent.

Sudo service datadog-agent restart (In Ubuntu)

