# Datadog Essentials

Rajesh Kumar

www.DevOpsSchool.com

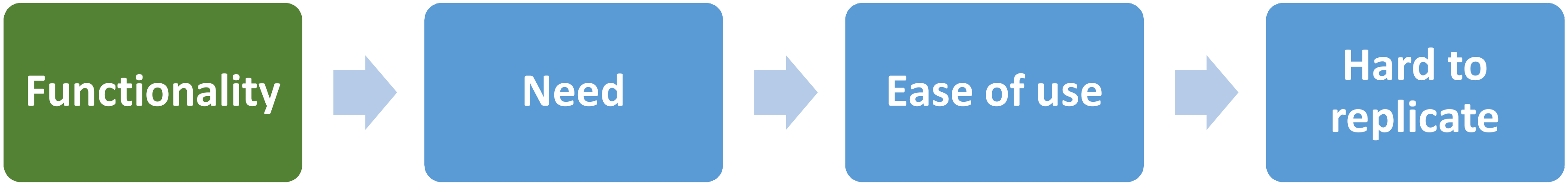
# What is Datadog?

- Datadog is a monitoring service for cloud-scale applications, providing monitoring of servers, databases, tools, and services, through a SaaS-based data analytics platform.
- They built Datadog to be a cloud infrastructure monitoring service, with a dashboard, alerting, and visualizations of metrics.
- Datadog was listed in Forbes’ Cloud 100 and was ranked in the top ten fastest growing companies in North America in Deloitte's 2016 Fast 500 List.
- Datadog was founded in 2010 by Olivier Pomel and Alexis Lê-Quôc

# Why  Datadog?

# Why Datadog?

- Providing functionality
- We need
- In an easy-to-use manner
- That would be difficult to build and maintain ourselves



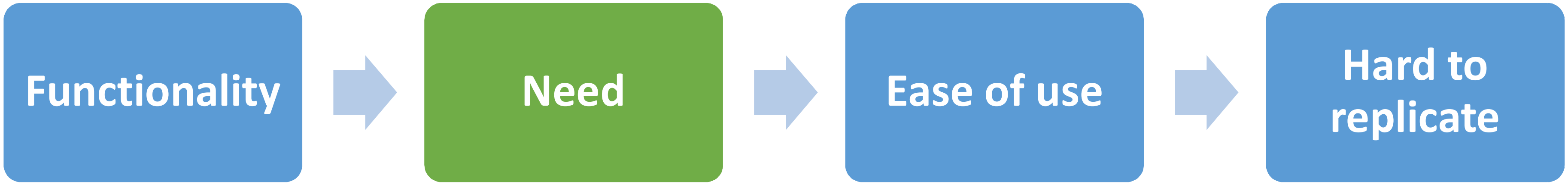
**The agent -** It gathers system metrics, integrates with key software we use, and provides a standard interface to which our applications can send custom metrics.

**Integrations -** Datadog has prebuilt integrations to pull data from almost every important service we use.

**Events -** Through the integrations datadog generates a consolidated event stream that we can filter and earch as needed.

**Dashboards -** Datadog lets us build dashboards that combine metrics from many different sources. We can combine and transform metrics to make them more useful. It also provides an powerful interface for interactive exploration of metrics.

**Alerting -** Datadog has nice stream processing capabilities for generating alerts, and it can surface them in services we use like pagerduty and slack.



Functionality → Need → Ease of use → Hard to replicate

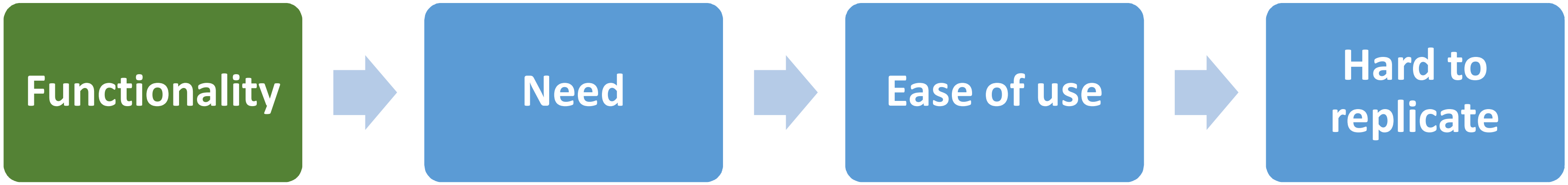
The Agent - We don't get nearly enough insight from cloudwatch alone, we need an on-instance tool to gather system and app metrics.

Integrations - There are lots of services with operational signficiance, but many of them don't provide a good way to access their data.

Events - We would spend dramatically longer investigating problems if we had to look at eash source of events in isolation. Many of our event sources don't even provide a way for us to view past events or to query them.

Dashboards - Per-service and per-instance dashboards are important for investigating problems quickly. The consolidation of data from multiple sources is again a key feature.

Alerting - We need to do anaylze trends in our metrics and alert on them.



Functionality

Need

Ease of use

Hard to replicate

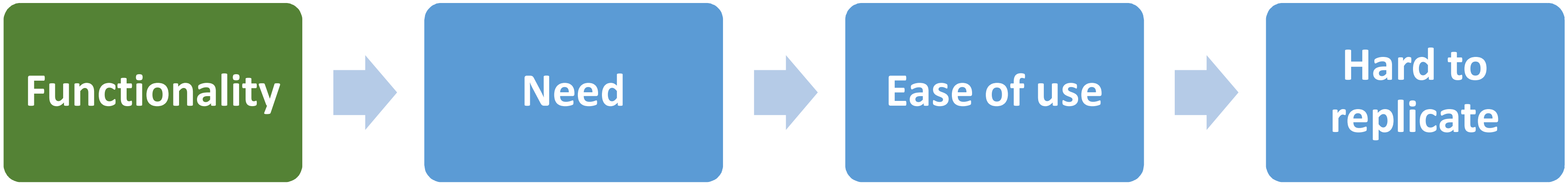
The agent - The agent is deployable via a chef cookbook datadog wrote for us. It requires minimal configuration. It knows which system and application metrics are worth gathering.

Integrations- Integrating with all the data sources is literally a few clicks.

Events - The interface makes searching and filtering events straightforward.

Dashboards - There are prebuilt dashbaords for lots of things we care about. Snazzy features like autocomplete and templating make building our own dashboards easy.

Alerting - The guided steps and previewed outputs make creating alerts simple.



Functionality

Need

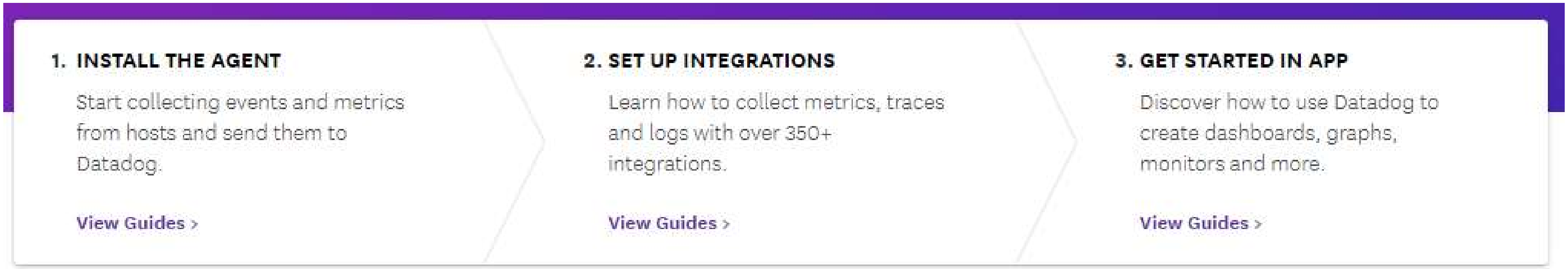
Ease of use

Hard to replicate

Here I described a system of collectd, custom code to pull metrics from cloudwatch, custom code to pull or receive events from various sources (airbrake, cloudtrail, chef, pagerduty, jenkins, etc) influxdb, and grafana.

|  | Is it Up? | Is it Slow? | Is it Broken? |
|---|---|---|---|
|  | Availability | Performance | Reliability |
| Application | Dynatrace<br>New Relic<br>Datadog | Dynatrace<br>New Relic<br>Datadog | Datadog |
| Network | Juniper<br>Datadog<br>Solar Winds | NetScout<br>Reverbed<br>Datadog | Datadog |
| Server | Nagios<br>NewRelic<br>Datadog | AppDynamics<br>NewRelic<br>Datadog | Datadog |

# Datadog Workflow



1. **INSTALL THE AGENT**

   Start collecting events and metrics from hosts and send them to Datadog.

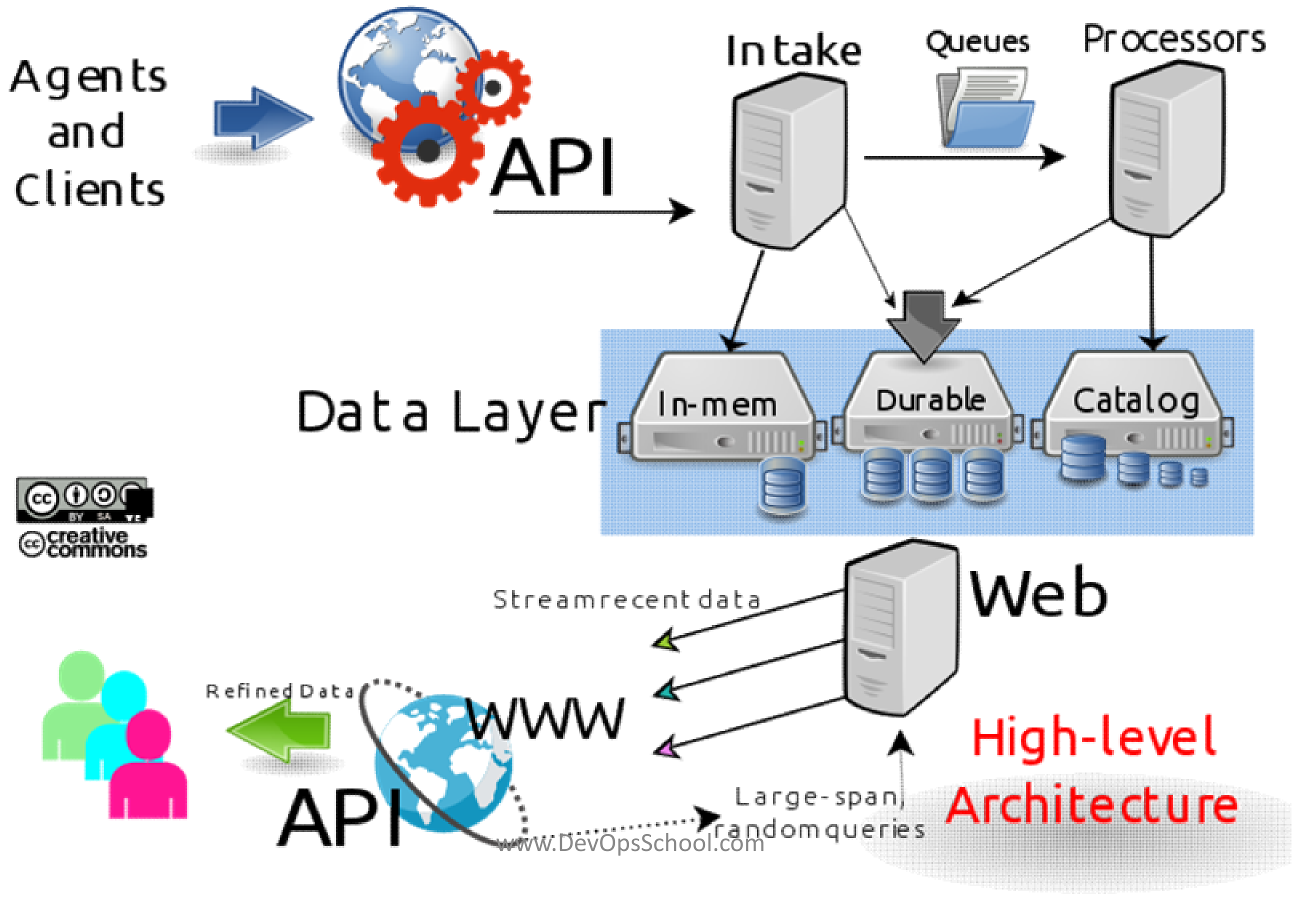
2. **SET UP INTEGRATIONS**

   Learn how to collect metrics, traces and logs with over 350+ integrations.

3. **GET STARTED IN APP**

   Discover how to use Datadog to create dashboards, graphs, monitors and more.

# Datadog high-level architecture



# Datadog high-level architecture

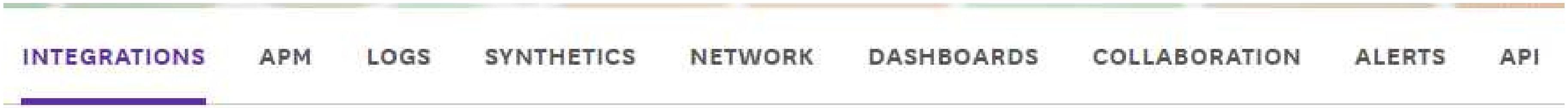
## Datadog Backend

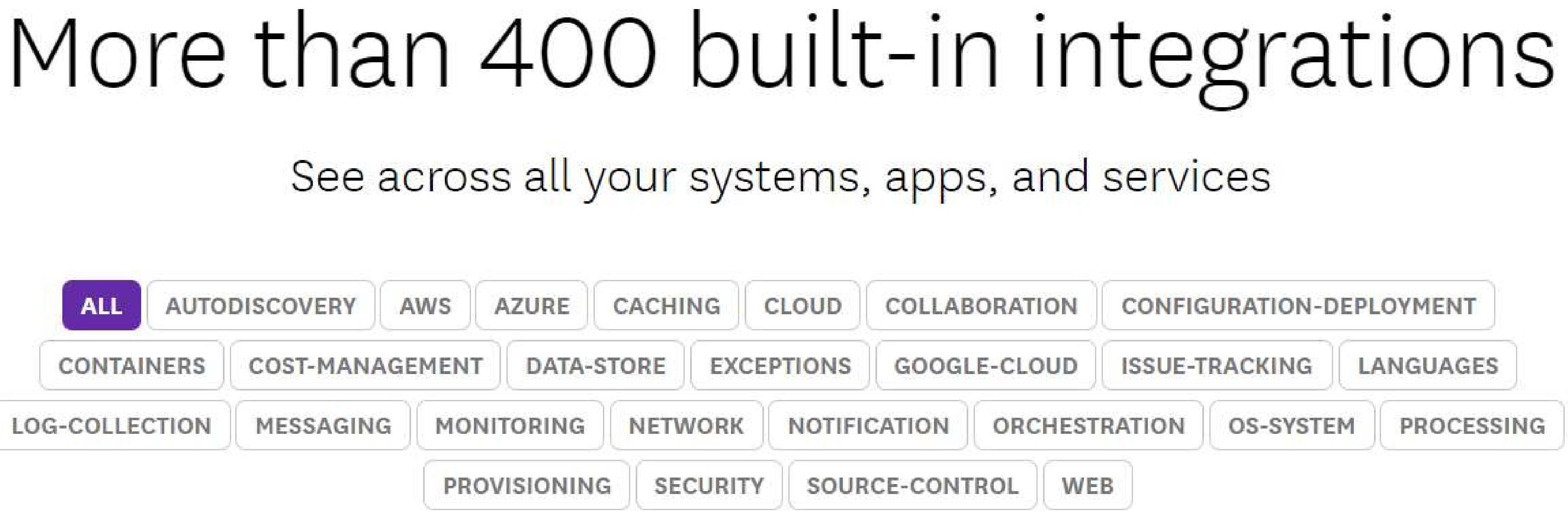
backend is built using a number of open and closed source technologies including D3, Apache Cassandra, Kafka, PostgreSQL, etc.

## Datadog Agent

Datadog uses a Go based agent, rewritten from scratch since its major version 6.0.0 released on February 28, 2018. It was formerly Python based.

# Features

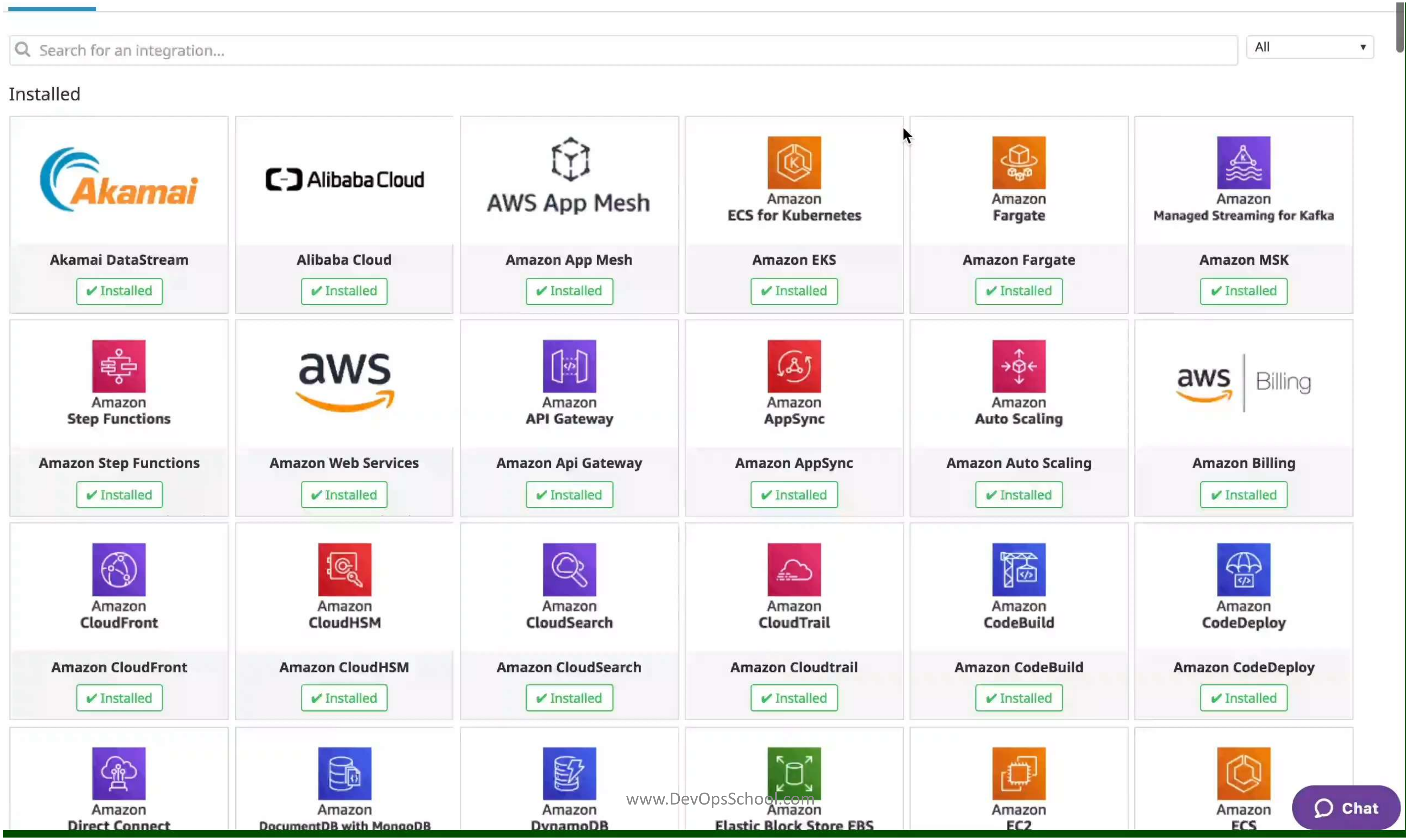

INTEGRATIONS
APM
LOGS
SYNTHETICS
NETWORK
DASHBOARDS
COLLABORATION
ALERTS
API



# More than 400 built-in integrations

See across all your systems, apps, and services

https://www.datadoghq.com/product/integrations/#all



Installed

Akamai DataStream ✔ Installed

Alibaba Cloud ✔ Installed

Amazon App Mesh ✔ Installed

Amazon EKS ✔ Installed

Amazon Fargate ✔ Installed

Amazon MSK ✔ Installed

Amazon Step Functions ✔ Installed

Amazon Web Services ✔ Installed

Amazon Api Gateway ✔ Installed

Amazon AppSync ✔ Installed

Amazon Auto Scaling ✔ Installed

Amazon Billing ✔ Installed

Amazon CloudFront ✔ Installed

Amazon CloudHSM ✔ Installed

Amazon CloudSearch ✔ Installed

Amazon Cloudtrail ✔ Installed

Amazon CodeBuild ✔ Installed

Amazon CodeDeploy ✔ Installed

# Real-time interactive dashboards

Correlate metrics, traces, logs, and more for collaborative analysis

# Unified log management

Monitor logs, metrics, and request traces in one platform for full-stack visibility

# Next-generation APM

Monitor, troubleshoot, and optimize end-to-end application performance

# Network Performance Monitoring

Visualize traffic flow in cloud-native environments

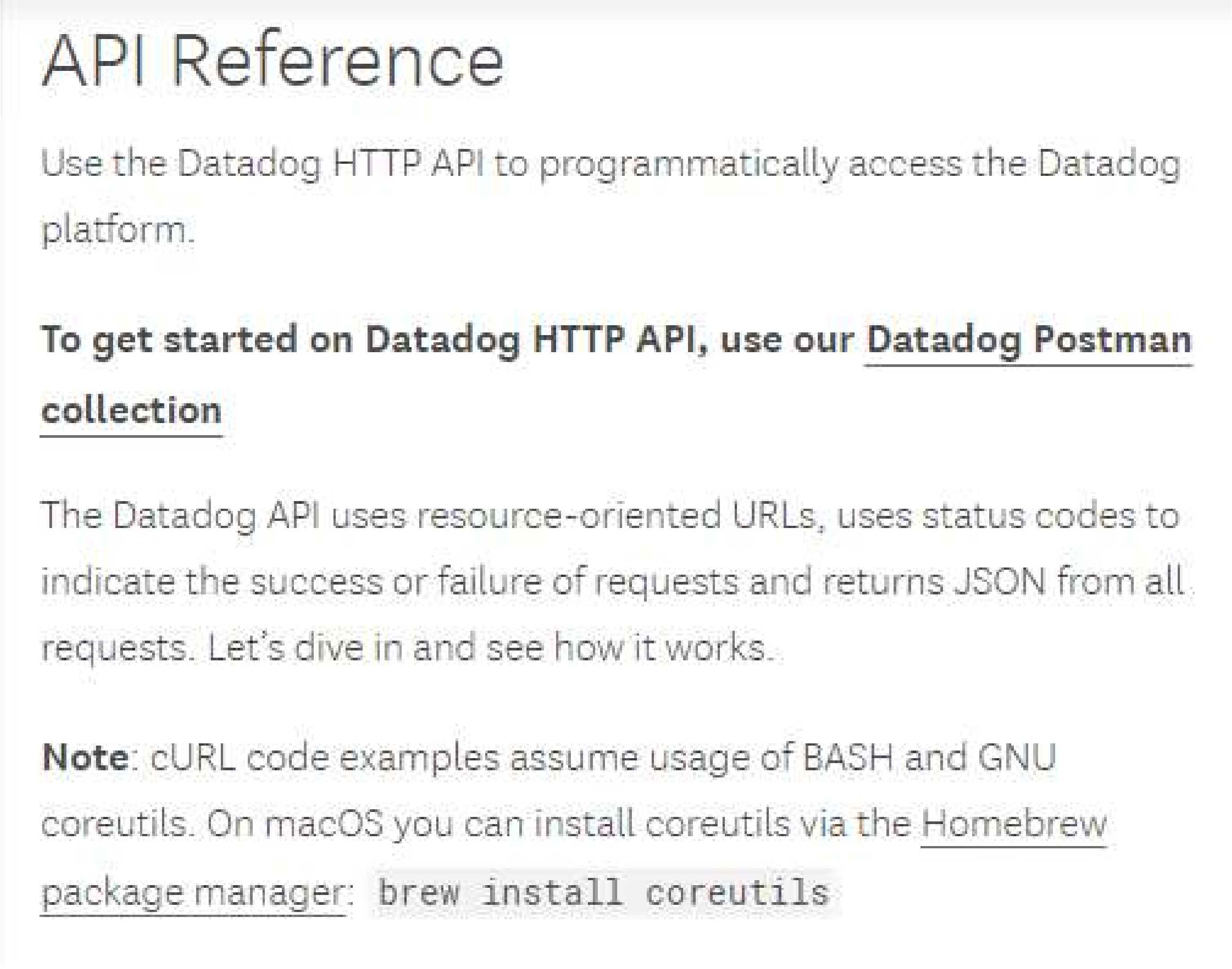
Simplified Synthetic Monitoring
End-to-end visibility across the stack in a single platform

# Real User Monitoring

End-to-end visibility into user journeys throughout the tech stack

# Actionable alerting

Eliminate false positives and receive actionable alerts driven by ML



# API Reference

Use the Datadog HTTP API to programmatically access the Datadog platform.

**To get started on Datadog HTTP API, use our Datadog Postman collection**

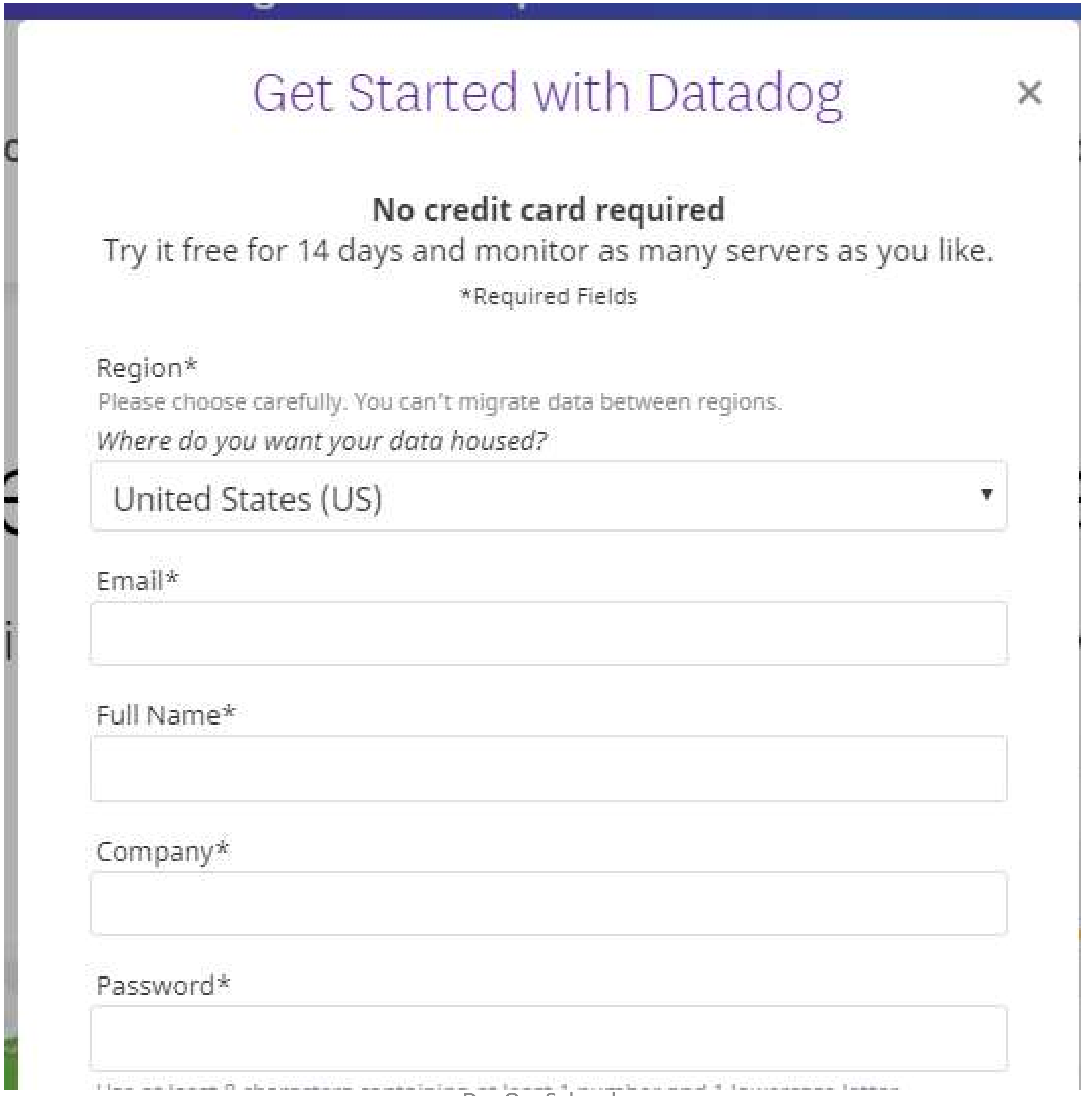
The Datadog API uses resource-oriented URLs, uses status codes to indicate the success or failure of requests and returns JSON from all requests. Let's dive in and see how it works.

**Note**: cURL code examples assume usage of BASH and GNU coreutils. On macOS you can install coreutils via the Homebrew package manager: `brew install coreutils`

# Getting Started

Create a Datadog accoun. Free for 14 days

https://www.datadoghq.com/



## Get Started with Datadog

**No credit card required**

Try it free for 14 days and monitor as many servers as you like.

*Required Fields

Region*

Please choose carefully. You can't migrate data between regions.

*Where do you want your data housed?*

United States (US) ▾

Email*

Full Name*

Company*

Password*

Use at least 8 characters containing at least 1 number and 1 lowercase letter

# Datadog terminology

- Agent
- DogStatsD
- StatsD
- metrics

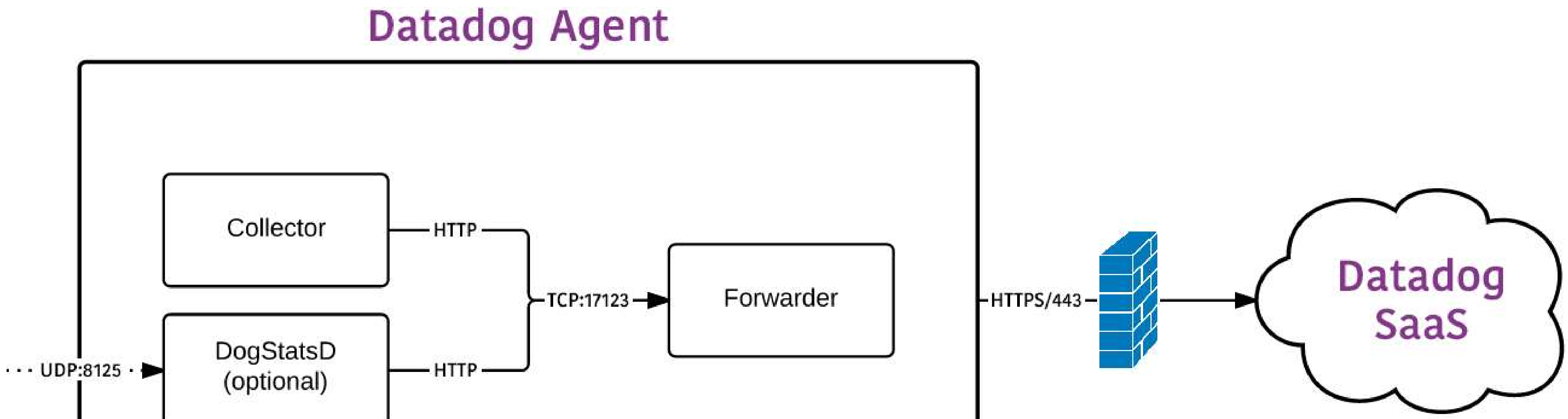
# Datadog Agent

The Agent is lightweight software installed on your hosts. It reports metrics and events from your host to Datadog via….

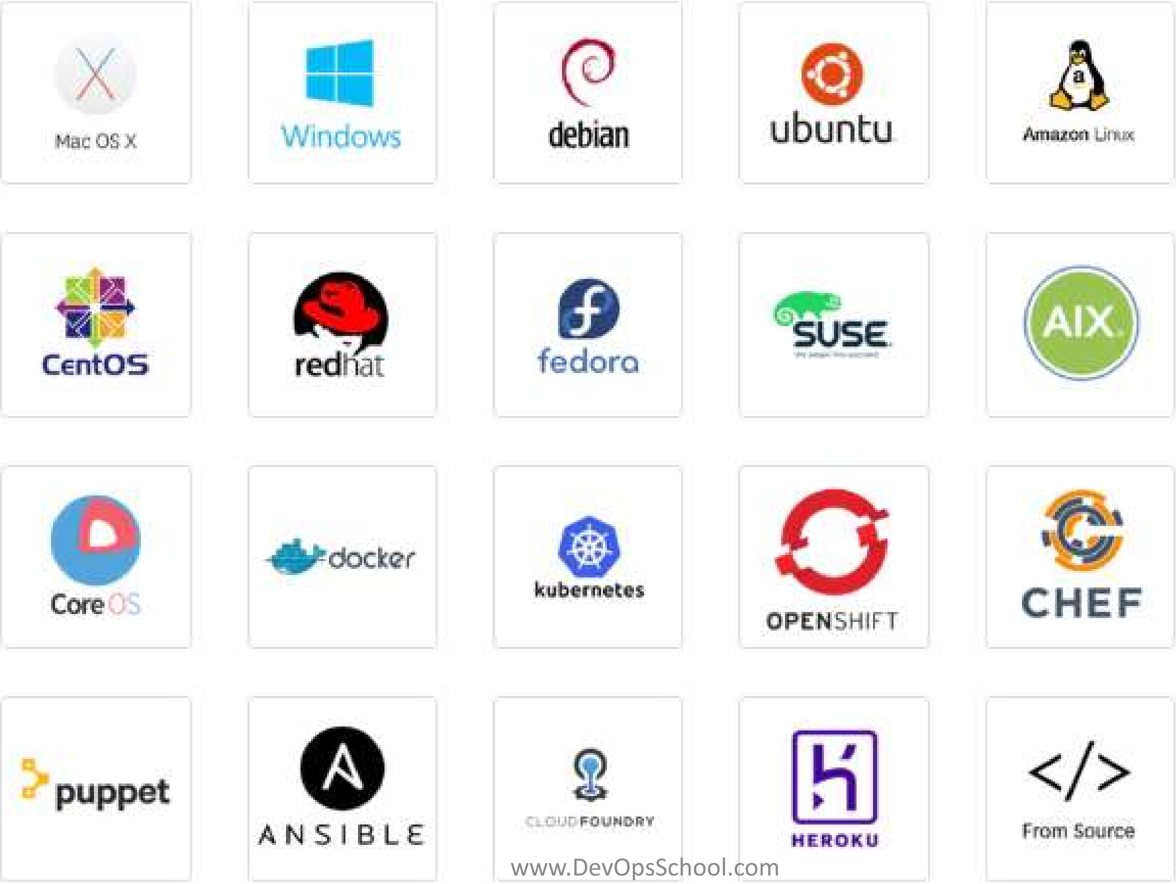
- **Integrations**, https://docs.datadoghq.com/integrations
- **DogStatsD**, or
- **The API.** https://docs.datadoghq.com/api/

# With additional setup, the Agent can report live processes, logs, and traces.

# How Datadog agent works?



# Datadog Agent: Platforms



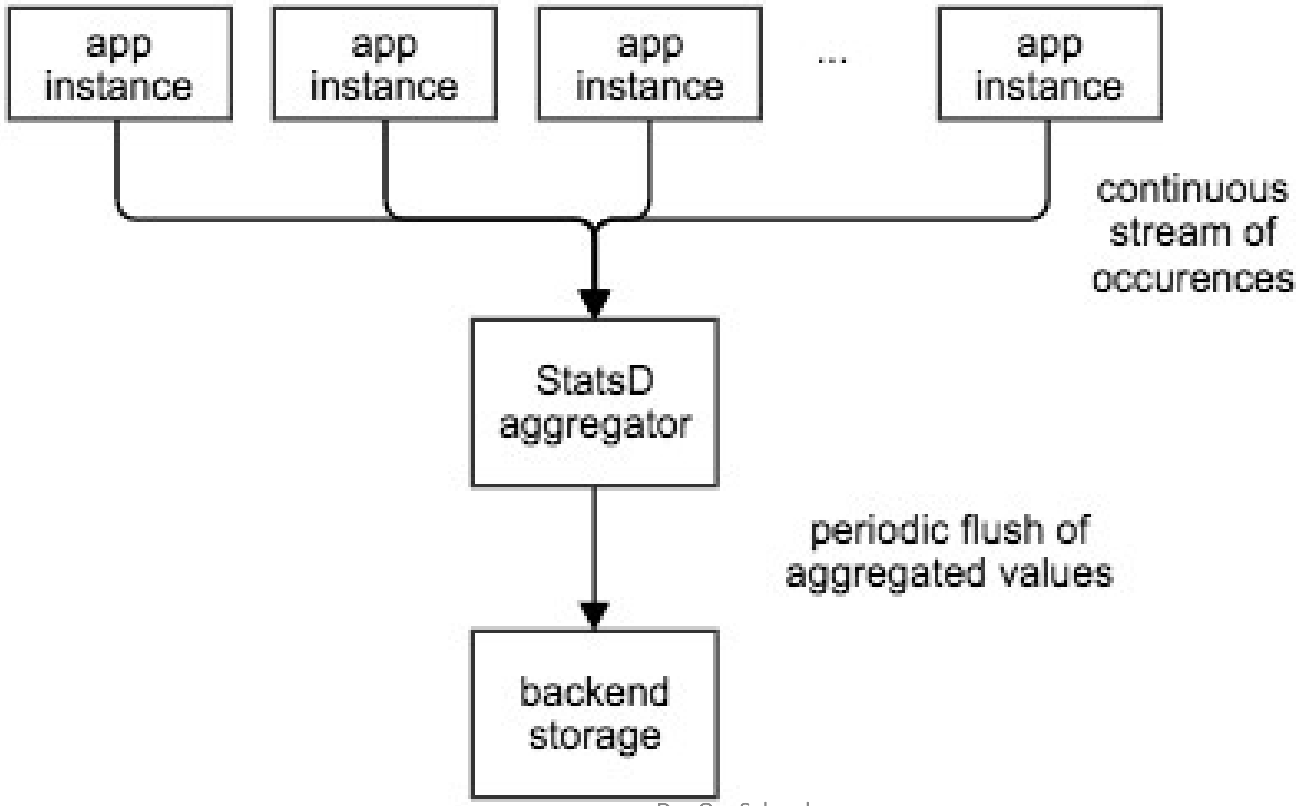
# StatsD

- StatsD is originally a simple daemon developed and released by Etsy to aggregate and summarize application metrics.
- With StatsD, applications are to be instrumented by developers using language-specific client libraries. These libraries will then communicate with the StatsD daemon using its dead-simple protocol, and the daemon will then generate aggregate metrics and relay them to virtually any graphing or monitoring backend.

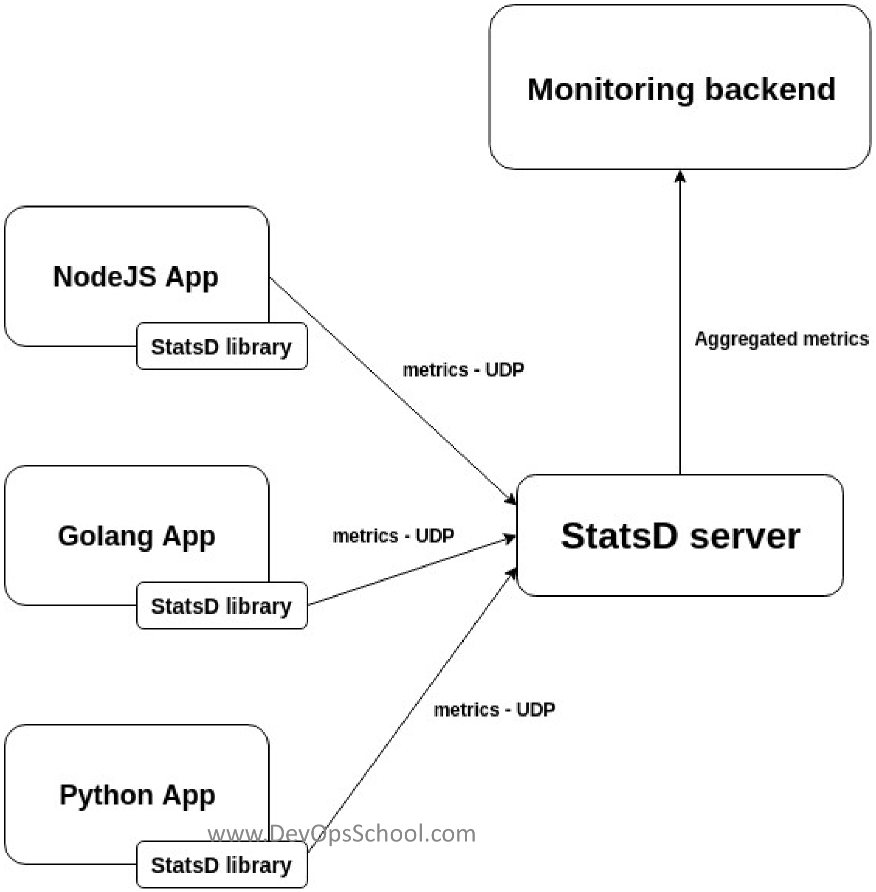
# How does StatsD work?

1. It all starts in your own application code. You—the developer—instrument it with one of the many StatsD libraries corresponding to  your app language. StatsD allows you to capture different types of metrics depending on your needs: today those are Gauges, Counters, Timing Summary Statistics, and Sets.
2. The StatsD client library then sends each individual call to the StatsD server over a UDP datagram. Since UDP is a disconnected protocol in which the recipient of a datagram doesn’t send any acknowledgement to the sender, the library doesn’t need to block when submitting data as it would with TCP or HTTP-based protocols.
3. The StatsD daemon will then listen to the UDP traffic from all application libraries, aggregate data over time and “flush” it at the desired interval to the backend of your choice. For example, individual function call timings may be aggregated every 10 seconds into a set of summary metrics describing its minimum, maximum, median, 90th and 95th percentile over the 10s interval. The protocol used  between the StatsD Daemon and the backend will vary depending on the backend used (most are HTTP-based).
4. The monitoring backend will turn your metrics from a stream of numbers on the wire into usable charts and alert you when needed. Examples of backends include tools like Graphite as well as yours truly.

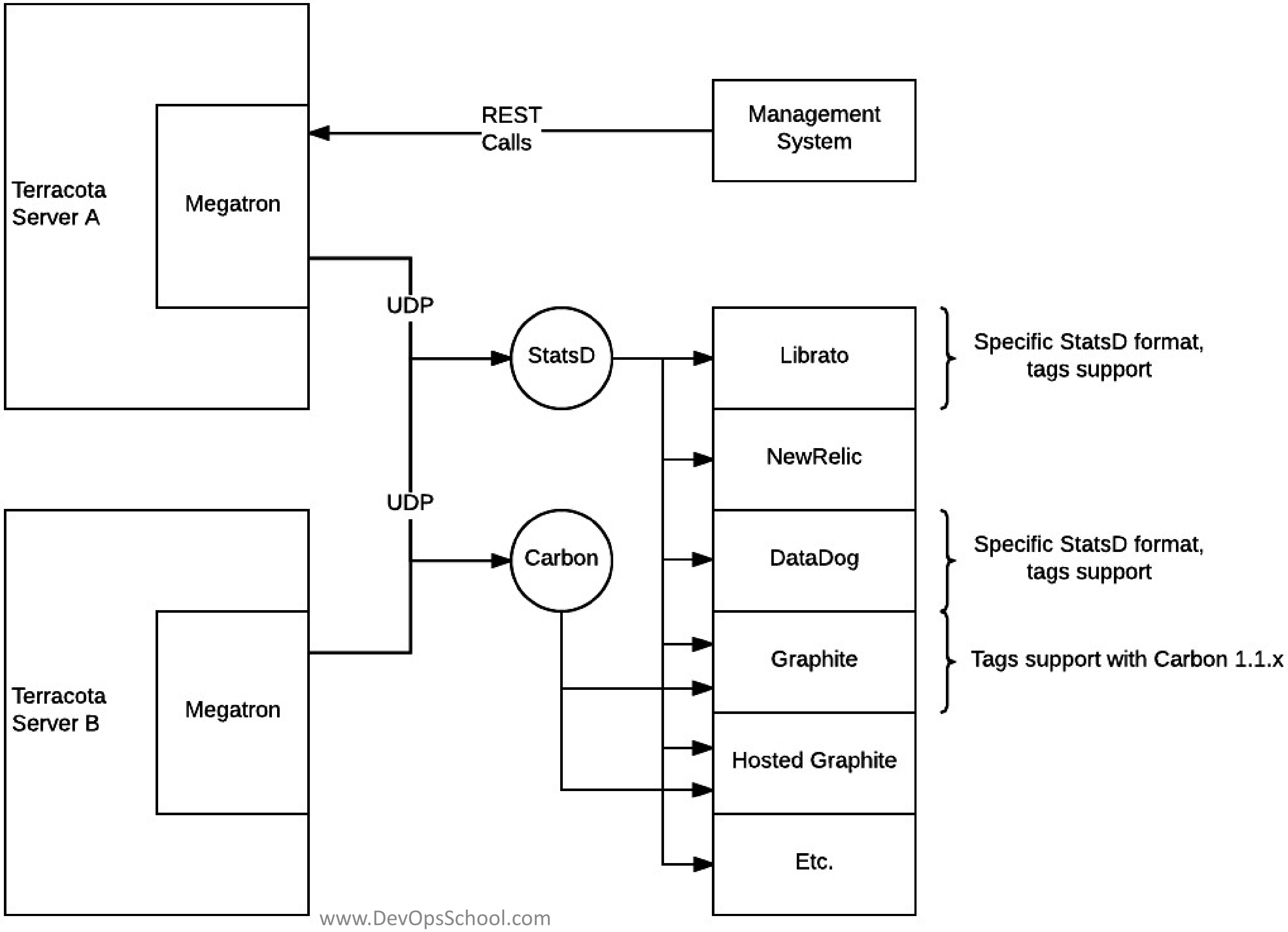
# How does StatsD work?



# How does StatsD work?



# How does StatsD work?

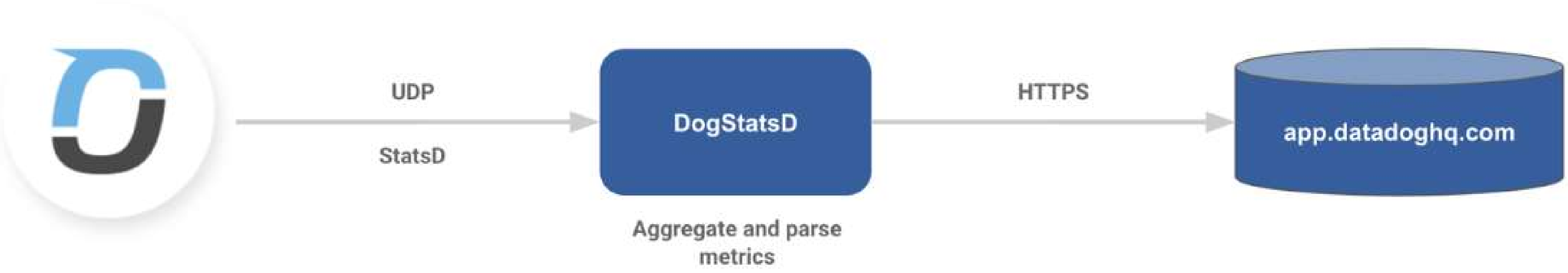


# StatsD & Datadog

- Datadog is big fans of StatsD and use it extensively internally. Customers to submit metrics from StatsD into Datadog for graphing, alerting, event correlation, and team collaboration: Datadog embedded our own StatsD daemon within the Datadog Agent, to make the setup as simple as possible
- Datadog extended the StatsD protocol to support tagging, one of Datadog’s killer features. This lets you add additional dimensions to your metrics, such as the application version, or type of customer a specific call relates to. But, we’ll come back to this in another post.
- Datadog made it very easy to discover StatsD metrics in the Datadog UI. Every host will automatically advertise its metrics, so you don’t have to look for them.

# Datadog Agent: DogStatsD

- While StatsD accepts only metrics, DogStatsD accepts all three of the major Datadog data types: metrics, events, and service checks.



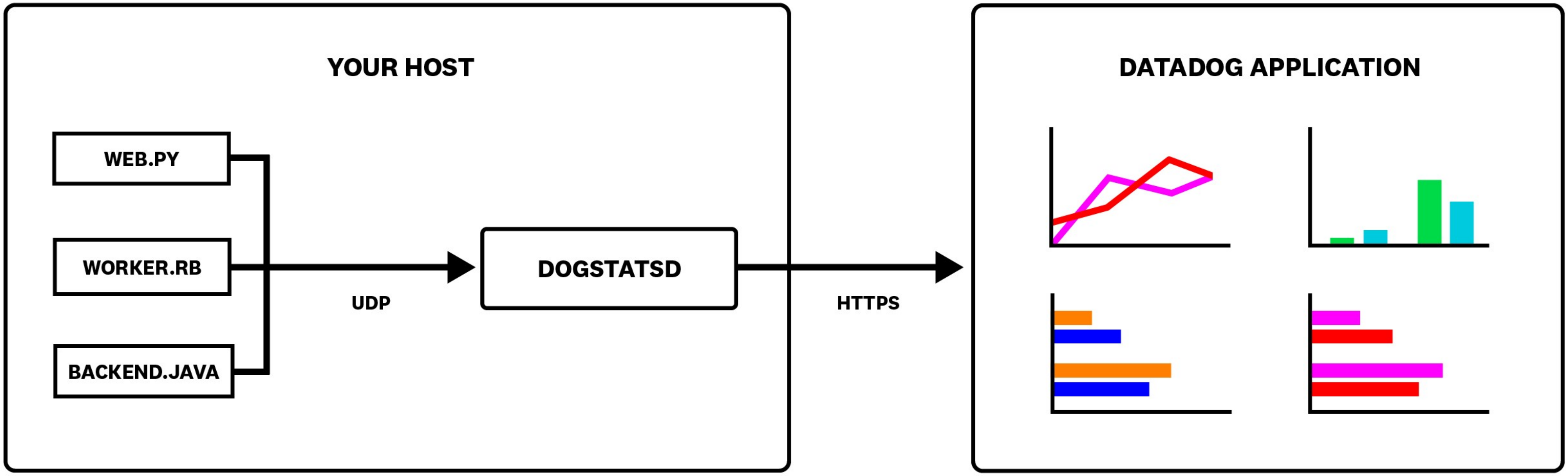
# Datadog Agent: DogStatsD

In v6, DogStatsD is a Golang implementation of Etsy’s StatsD metric aggregation daemon. It is used to receive and roll up arbitrary metrics over UDP or Unix socket, thus allowing custom code to be instrumented without adding latency

# How does DogStatsD work?

- DogStatsD accepts custom metrics, events, and service checks over UDP and periodically aggregates and forwards them to Datadog.
- Because it uses UDP, your application can send metrics to DogStatsD and resume its work without waiting for a response. If DogStatsD ever becomes unavailable, your application won’t experience an interruption.
- As it receives data, DogStatsD aggregates multiple data points for each unique metric into a single data point over a period of time called the flush interval (ten seconds, by default).

# How does DogStatsD work?



# Datadog vs StatsD: What are the differences?

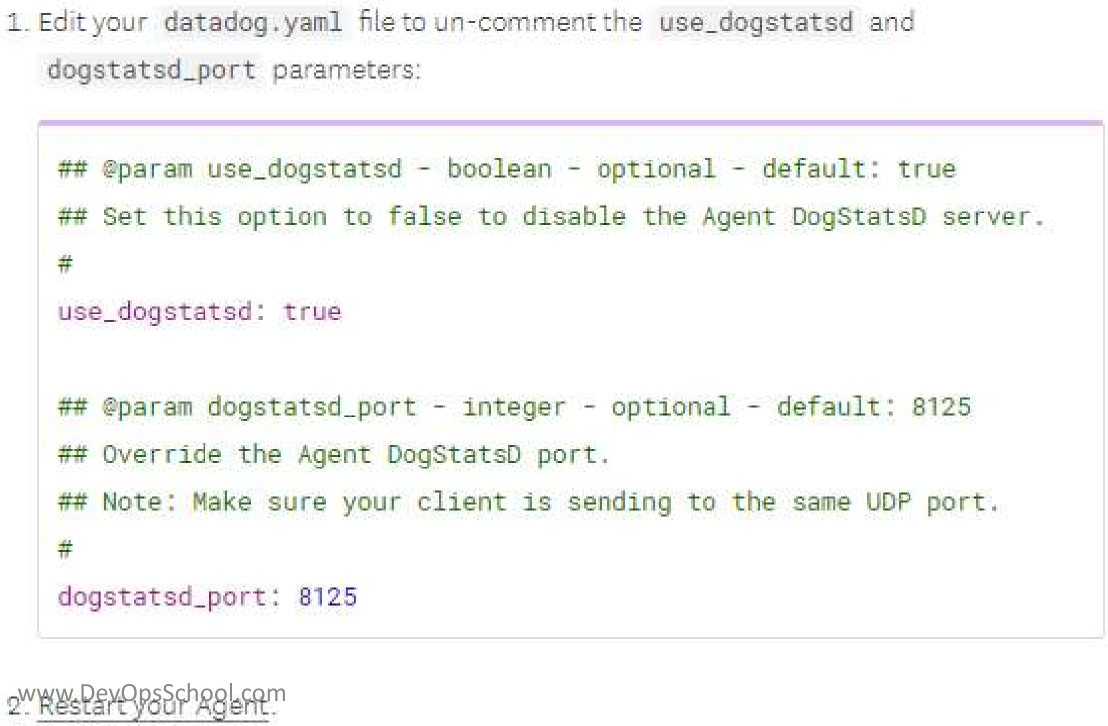
The easiest way to get your custom application metrics into Datadog is to send them to DogStatsD, a metrics aggregation service bundled with the Datadog Agent. DogStatsD implements the StatsD protocol and adds a few Datadog-specific extensions:

- Histogram metric type
- Service checks
- Events
- Tagging

Any compliant StatsD client works with DogStatsD and the Agent, but you won’t be able to use the Datadog-specific extensions.

# Datadog Agent: DogStatsD

DogStatsD is enabled by default over UDP port 8125 for Agent v6+. By default, DogStatsD listens on UDP port 8125.  You can also configure DogStatsD to use a Unix domain socket. To enable a custom Agent DogStatsD server UDP port:



1. Edit your `datadog.yaml` file to un-comment the `use_dogstatsd` and `dogstatsd_port` parameters:

```
## @param use_dogstatsd - boolean - optional - default: true
## Set this option to false to disable the Agent DogStatsD server.
#
use_dogstatsd: true

## @param dogstatsd_port - integer - optional - default: 8125
## Override the Agent DogStatsD port.
## Note: Make sure your client is sending to the same UDP port.
#
dogstatsd_port: 8125
```

2. Restart your Agent.

# Datadog Agent: The Collector

The collector gathers all standard metrics every 15 seconds. Agent v6 embeds a Python 2.7 interpreter to run integrations and custom checks.

# Datadog Agent: The Forwarder

The Agent forwarder send metrics over HTTPS to Datadog. Buffering prevents network splits from affecting metric reporting.

Metrics are buffered in memory until a limit in size or number of outstanding send requests are reached.

Afterwards, the oldest metrics are discarded to keep the forwarder’s memory footprint manageable. Logs are sent over an SSL-encrypted TCP connection to Datadog.

# Agent v5

Agent v5 is composed of four major components, each written in Python running as a separate process:

- Collector (agent.py): The collector runs checks on the current machine for configured integrations, and captures system metrics, such as memory and CPU.
- DogStatsD (dogstatsd.py): This is a StatsD-compatible backend server that you can send custom metrics to from your applications.
- Forwarder (ddagent.py): The forwarder retrieves data from both DogStatsD and the collector, queues it up, and then sends it to Datadog.
- SupervisorD: This is all controlled by a single supervisor process. It is kept separate to limit the overhead of each application if you aren’t running all parts. However, it is generally recommended to run all parts.

# Agent v6 & v7

Agent v6 and v7 are composed of a main process responsible for collecting infrastructure metrics, logs, and receiving DogStatsD metrics. The main components to this process are:

- The Collector is in charge of running checks and collecting metrics.
- The Forwarder sends payloads to Datadog.

# Agent v6 & v7

Two optional processes are spawned by the Agent if enabled in the datadog.yaml configuration file:

- The APM Agent is a process to collect traces (enabled by default).
- The Process Agent is a process to collect live process information. By default, it only collects available containers, otherwise it is disabled.

# Datadog Agent: Agent GUI

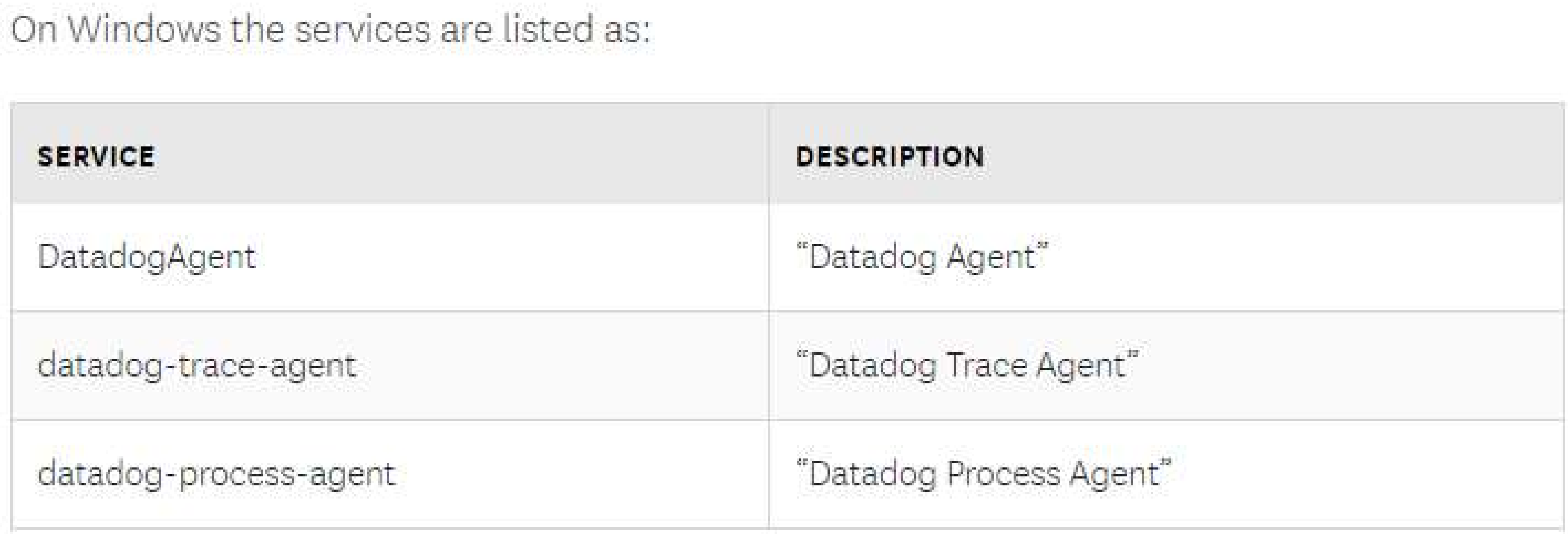
You can configure the port on which the GUI runs in the datadog.yaml file. To disable the GUI, set the port’s value to -1. For Windows and macOS, the GUI is enabled by default and runs on port 5002. For Linux, the GUI is disabled by default.

When the Agent is running, use the datadog-agent launch-gui command to open the GUI in your default web browser.

# Datadog Agent: Agent GUI: Requirements

- Cookies must be enabled in your browser. The GUI generates and saves a token in your browser which is used for authenticating all communications with the GUI server.
- To start the GUI, the user must have the required permissions. If you are able to open datadog.yaml, you are able to use the GUI.
- For security reasons, the GUI can only be accessed from the local network interface (localhost/127.0.0.1), therefore you must be on the same host that the Agent is running. That is, you can’t run the Agent on a VM or a container and access it from the host machine.

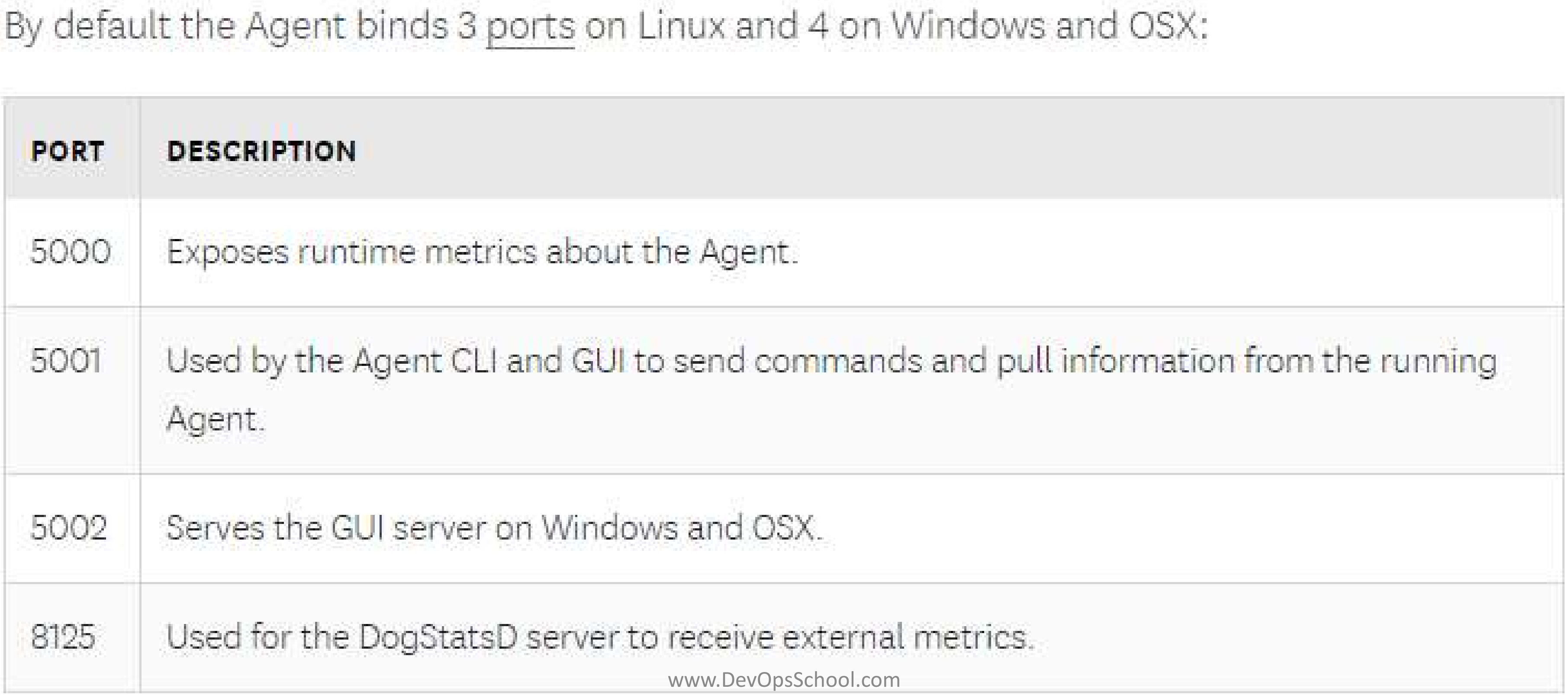
# Datadog Agent services in Windows



On Windows the services are listed as:

| SERVICE | DESCRIPTION |
|---|---|
| DatadogAgent | “Datadog Agent” |
| datadog-trace-agent | “Datadog Trace Agent” |
| datadog-process-agent | “Datadog Process Agent” |

# Datadog Agent Ports in Windows and OSX

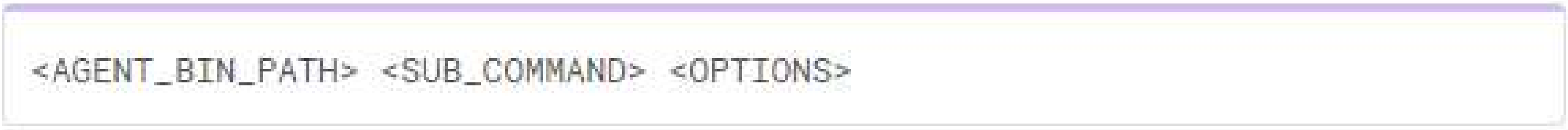


By default the Agent binds 3 ports on Linux and 4 on Windows and OSX:

| PORT | DESCRIPTION |
|---|---|
| 5000 | Exposes runtime metrics about the Agent. |
| 5001 | Used by the Agent CLI and GUI to send commands and pull information from the running Agent. |
| 5002 | Serves the GUI server on Windows and OSX. |
| 8125 | Used for the DogStatsD server to receive external metrics. |

# Datadog Agent: CLI

With Agent v6+, the command line interface is based on subcommands. To run a subcommand, first invoke the Agent binary:



```
<AGENT_BIN_PATH> <SUB_COMMAND> <OPTIONS>
```

# Datadog Agent: CLI

| SUBCOMMAND | NOTES |
|---|---|
| check | Run the specified check. |
| configcheck | Print all configurations loaded & resolved of a running Agent. |
| diagnose | Execute connectivity diagnosis on your system. |
| flare | Collect a flare and send it to Datadog. |
| health | Print the current Agent health. |
| help | Help about any command. |
| hostname | Print the hostname used by the Agent. |
| import | Import and convert configuration files from previous versions of the Agent. |

# Datadog Agent: CLI

| | |
|---|---|
| installservice | Install the Agent within the service control manager. |
| launch-gui | Start the Datadog Agent GUI. |
| regimport | Import the registry settings into datadog.yaml. |
| remove-service | Remove the Agent from the service control manager. |
| restart | Restart the Agent. |
| restart-service | Restart the Agent within the service control manager. |
| start | Start the Agent. |
| start-service | Start the Agent within the service control manager. |
| status | Print the current Agent status. |
| stop | Stop the Agent. |
| stopservice | Stop the Agent within the service control manager. |
| version | Print version info. |

# Datadog Agent: datadog.yaml

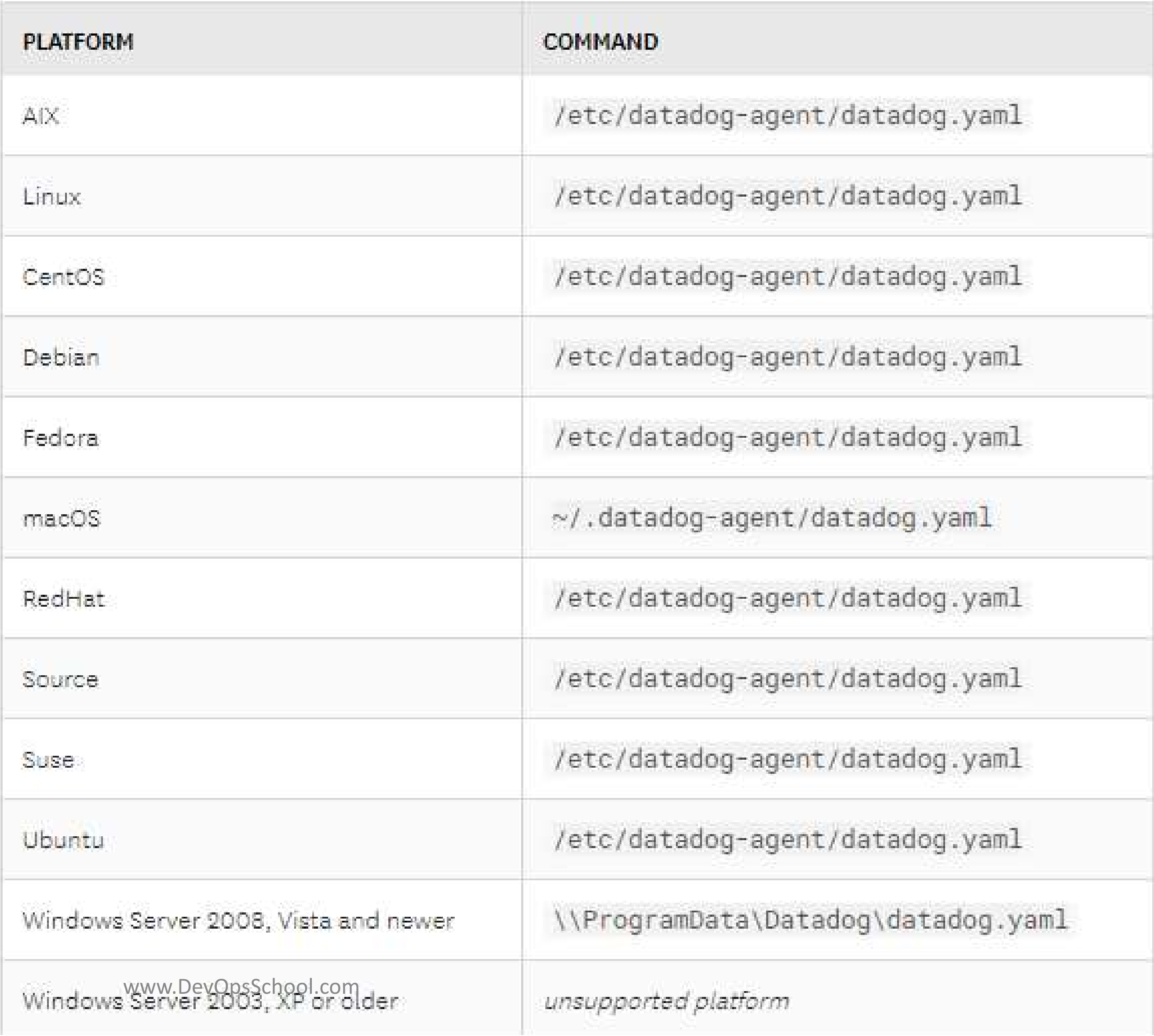
## Agent Configuration Files

- The Agent v6 configuration file uses YAML to better support complex configurations, and to provide a consistent configuration experience, as Checks also use YAML configuration files.

https://github.com/DataDog/datadog-agent/blob/master/pkg/config/config_template.yaml

# Datadog Agent: datadog.yaml

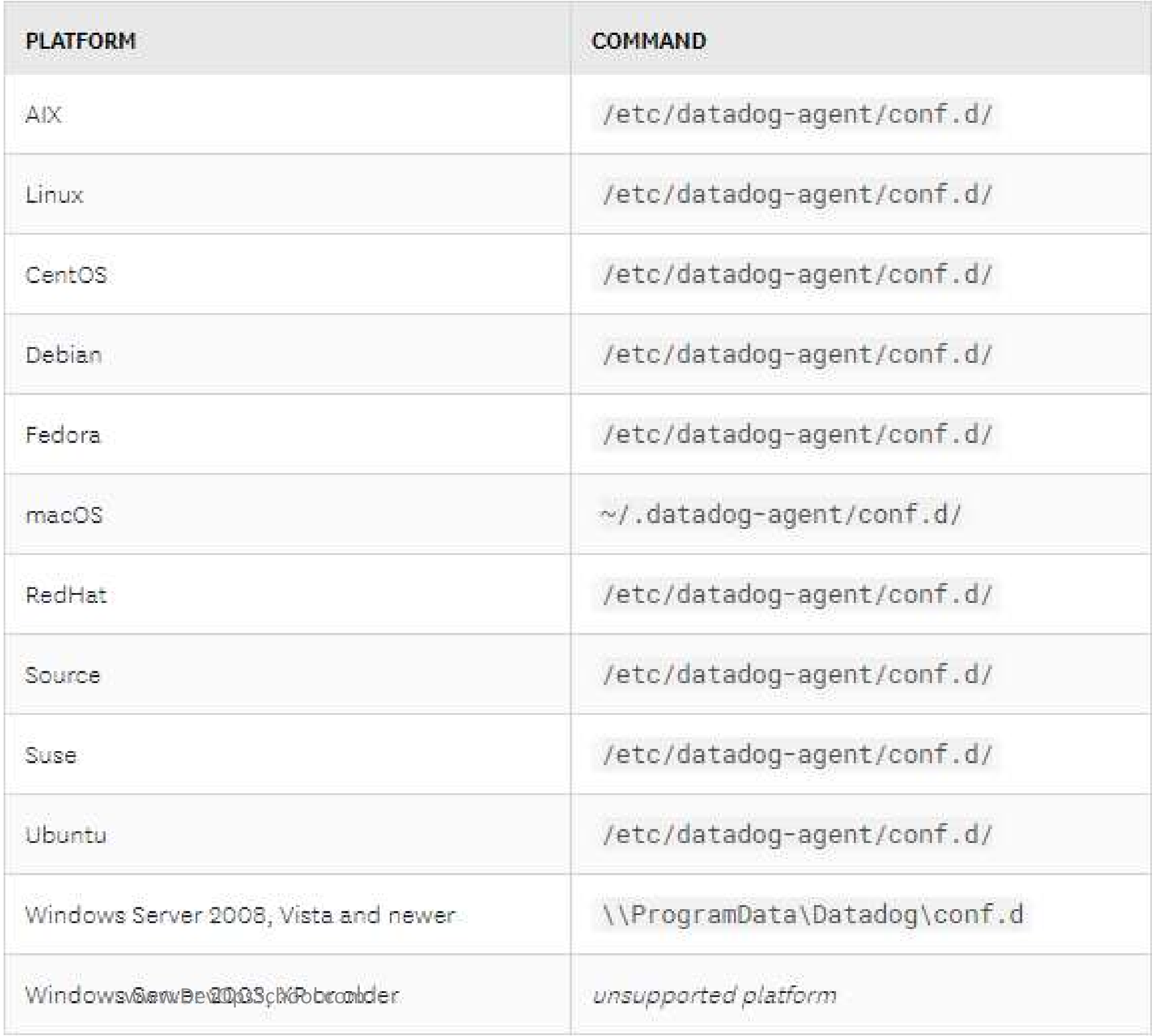
Agent Configuration Files



| PLATFORM | COMMAND |
|---|---|
| AIX | `/etc/datadog-agent/datadog.yaml` |
| Linux | `/etc/datadog-agent/datadog.yaml` |
| CentOS | `/etc/datadog-agent/datadog.yaml` |
| Debian | `/etc/datadog-agent/datadog.yaml` |
| Fedora | `/etc/datadog-agent/datadog.yaml` |
| macOS | `~/.datadog-agent/datadog.yaml` |
| RedHat | `/etc/datadog-agent/datadog.yaml` |
| Source | `/etc/datadog-agent/datadog.yaml` |
| Suse | `/etc/datadog-agent/datadog.yaml` |
| Ubuntu | `/etc/datadog-agent/datadog.yaml` |
| Windows Server 2008, Vista and newer | `\\ProgramData\Datadog\datadog.yaml` |
| Windows Server 2003, XP or older | *unsupported platform* |

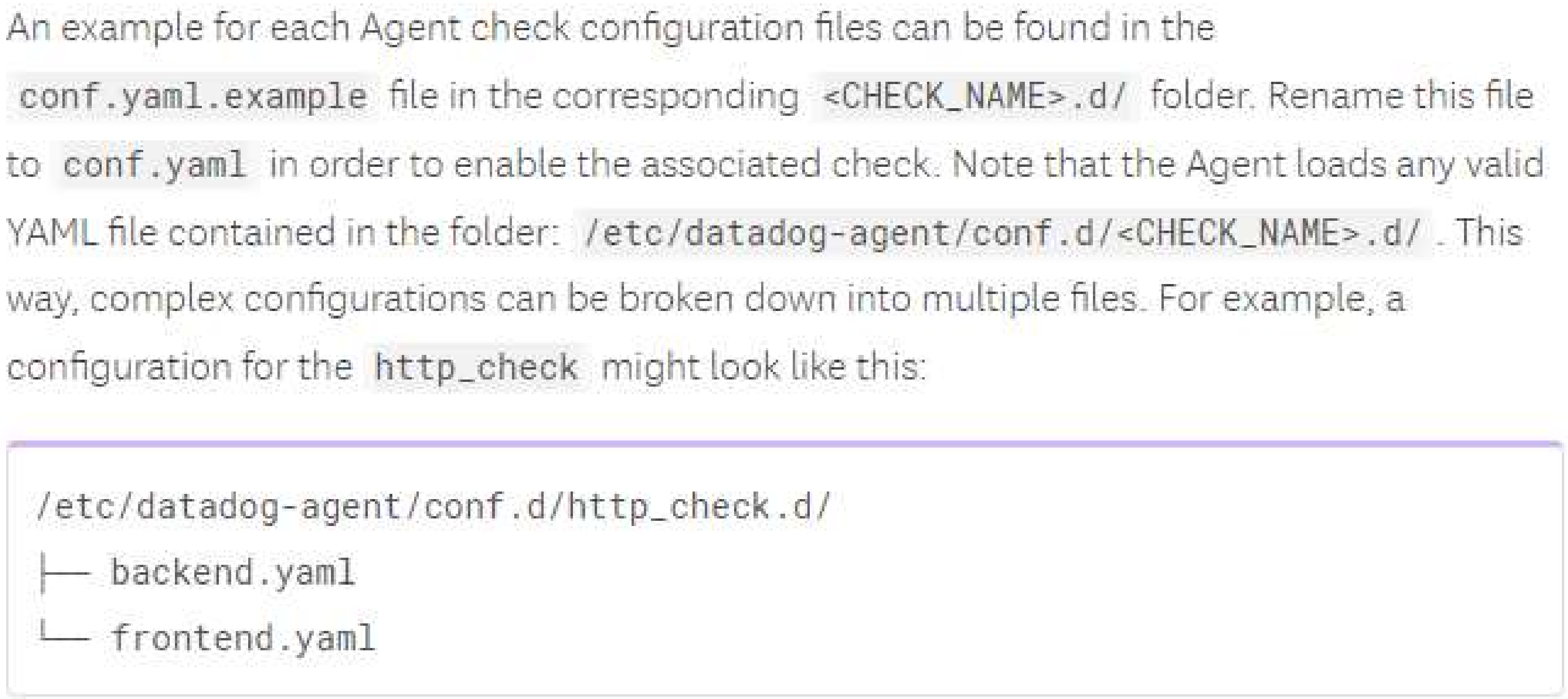
# Datadog Agent:

Agent configuration directory



| PLATFORM | COMMAND |
|---|---|
| AIX | `/etc/datadog-agent/conf.d/` |
| Linux | `/etc/datadog-agent/conf.d/` |
| CentOS | `/etc/datadog-agent/conf.d/` |
| Debian | `/etc/datadog-agent/conf.d/` |
| Fedora | `/etc/datadog-agent/conf.d/` |
| macOS | `~/.datadog-agent/conf.d/` |
| RedHat | `/etc/datadog-agent/conf.d/` |
| Source | `/etc/datadog-agent/conf.d/` |
| Suse | `/etc/datadog-agent/conf.d/` |
| Ubuntu | `/etc/datadog-agent/conf.d/` |
| Windows Server 2008, Vista and newer | `\\ProgramData\Datadog\conf.d` |
| Windows Server 2003, XP or older | *unsupported platform* |

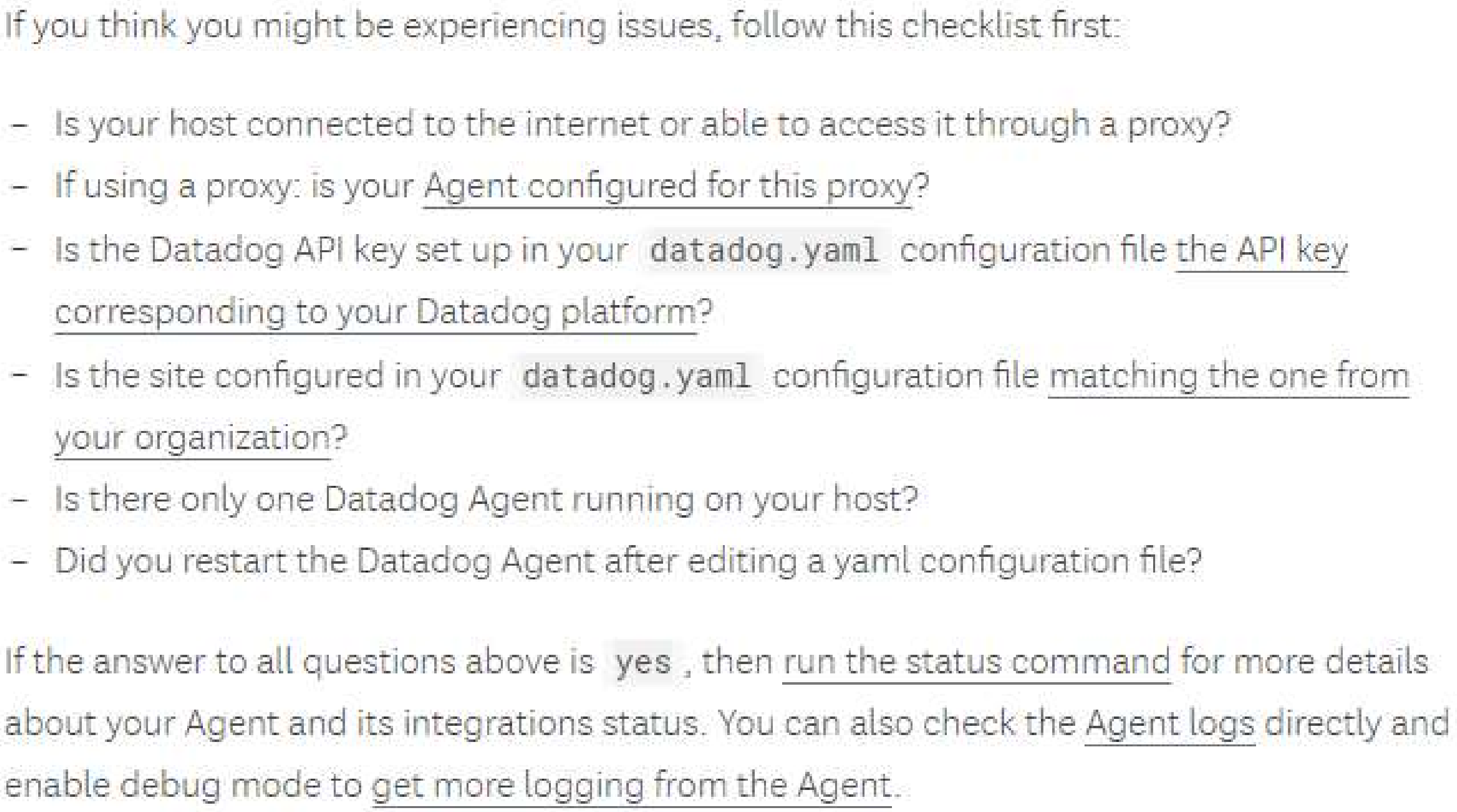
# How to Enabled Datadog Agent check configuration files?



An example for each Agent check configuration files can be found in the `conf.yaml.example` file in the corresponding `<CHECK_NAME>.d/` folder. Rename this file to `conf.yaml` in order to enable the associated check. Note that the Agent loads any valid YAML file contained in the folder: `/etc/datadog-agent/conf.d/<CHECK_NAME>.d/` . This way, complex configurations can be broken down into multiple files. For example, a configuration for the `http_check` might look like this:

```
/etc/datadog-agent/conf.d/http_check.d/
├── backend.yaml
└── frontend.yaml
```

# Datadog Agent: Troubleshooting



If you think you might be experiencing issues, follow this checklist first:

- Is your host connected to the internet or able to access it through a proxy?
- If using a proxy: is your Agent configured for this proxy?
- Is the Datadog API key set up in your `datadog.yaml` configuration file the API key corresponding to your Datadog platform?
- Is the site configured in your `datadog.yaml` configuration file matching the one from your organization?
- Is there only one Datadog Agent running on your host?
- Did you restart the Datadog Agent after editing a yaml configuration file?

If the answer to all questions above is `yes` , then run the status command for more details about your Agent and its integrations status. You can also check the Agent logs directly and enable debug mode to get more logging from the Agent.

https://docs.datadoghq.com/agent/troubleshooting/

# Datadog Agent: Start the Agent

| PLATFORM | COMMAND |
|---|---|
| AIX | startsrc -s datadog-agent |
| Linux | sudo service datadog-agent start |
| Docker | See the dedicated Docker documentation |
| Kubernetes | kubectl create -f datadog-agent.yaml |
| macOS | launchctl start com.datadoghq.agent *or* via the systray app |
| Source | sudo service datadog-agent start |
| Windows | See the dedicated Windows documentation |

# Datadog Agent: Start the Agent

| PLATFORM | COMMAND |
|---|---|
| AIX | stopsrc -s datadog-agent |
| Linux | sudo service datadog-agent stop |
| Docker | See the dedicated Docker documentation |
| Kubernetes | kubectl delete pod <AGENT POD NAME>—note: the pod is automatically rescheduled |
| macOS | launchctl stop com.datadoghq.agent *or* via the systray app |
| Source | sudo service datadog-agent stop |
| Windows | See the dedicated Windows documentation |

# Datadog Agent: Stop the Agent

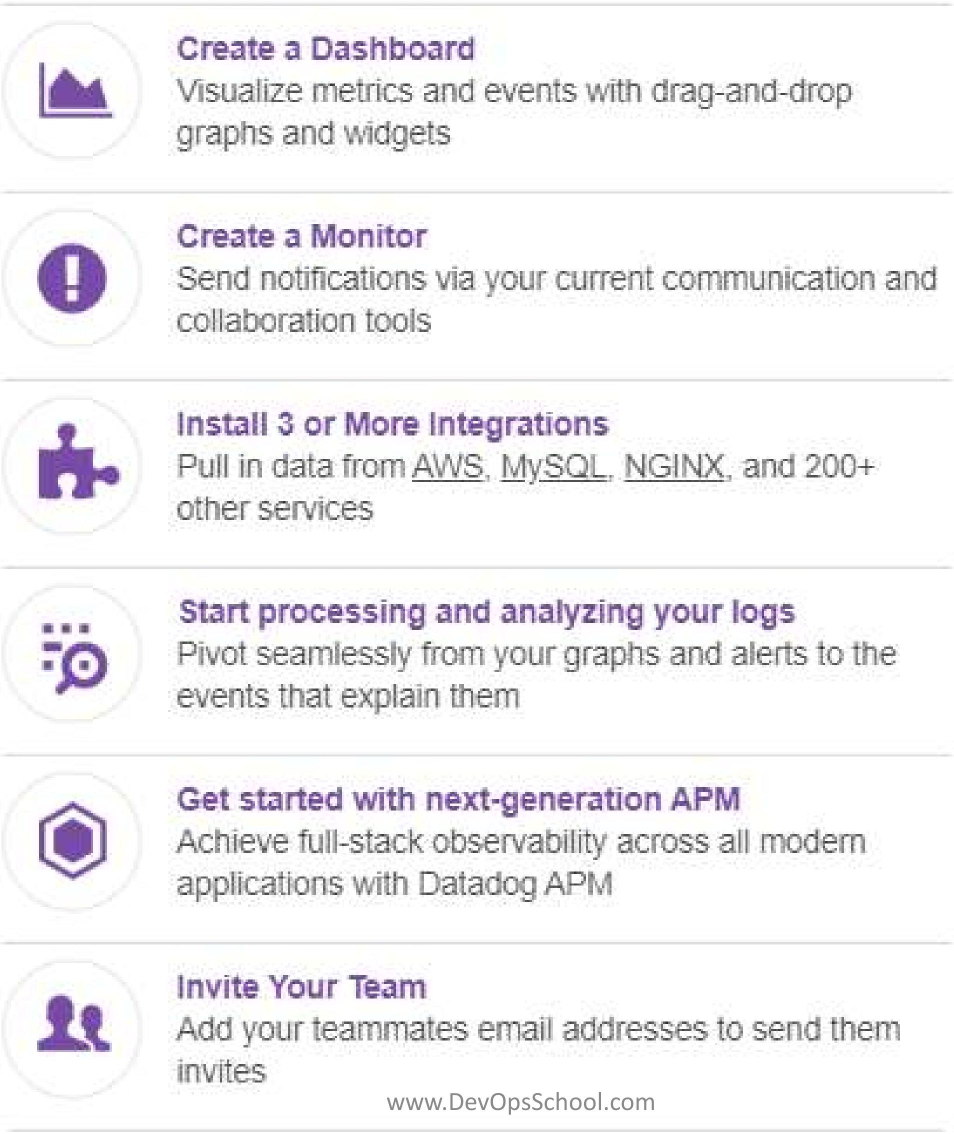
| PLATFORM | COMMAND |
|---|---|
| AIX | stopsrc -s datadog-agent |
| Linux | sudo service datadog-agent stop |
| Docker | See the dedicated Docker documentation |
| Kubernetes | kubectl delete pod <AGENT POD NAME>—note: the pod is automatically rescheduled |
| macOS | launchctl stop com.datadoghq.agent *or* via the systray app |
| Source | sudo service datadog-agent stop |
| Windows | See the dedicated Windows documentation |

# Datadog Agent: Service Status

| PLATFORM | COMMAND |
|---|---|
| AIX | lssrc -s datadog-agent |
| Linux | sudo service datadog-agent status |
| Docker (Debian) | sudo docker exec -it <CONTAINER_NAME> s6-svstat /var/run/s6/services/agent/ |
| Kubernetes | kubectl exec -it <POD_NAME> s6-svstat /var/run/s6/services/agent/ |
| macOS | launchctl list com.datadoghq.agent *or* via the systray app |
| Source | sudo service datadog-agent status |

# Datadog Agent: Agent Status

| PLATFORM | COMMAND |
|---|---|
| AIX | datadog-agent status |
| Linux | sudo datadog-agent status |
| Docker | sudo docker exec -it <CONTAINER_NAME> agent status |
| Kubernetes | kubectl exec -it <POD_NAME> agent status |
| macOS | datadog-agent status or via the web GUI |
| Source | sudo datadog-agent status |
| Windows | See the dedicated Windows documentation |



**Create a Dashboard**
Visualize metrics and events with drag-and-drop graphs and widgets

**Create a Monitor**
Send notifications via your current communication and collaboration tools

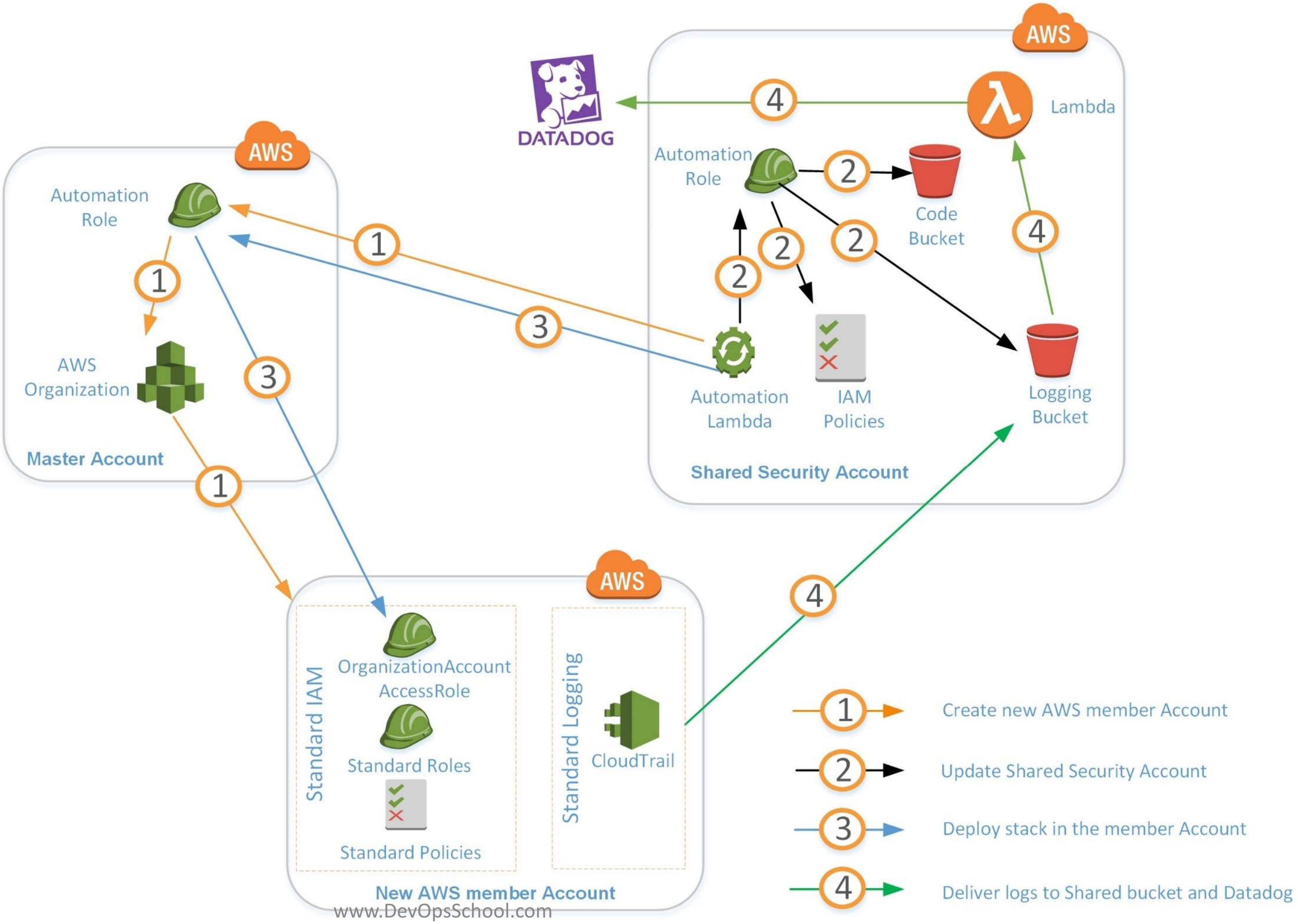
**Install 3 or More Integrations**
Pull in data from AWS, MySQL, NGINX, and 200+ other services

**Start processing and analyzing your logs**
Pivot seamlessly from your graphs and alerts to the events that explain them

**Get started with next-generation APM**
Achieve full-stack observability across all modern applications with Datadog APM

**Invite Your Team**
Add your teammates email addresses to send them invites

# Datadog with AWS



1 Create new AWS member Account

2 Update Shared Security Account

3 Deploy stack in the member Account

4 Deliver logs to Shared bucket and Datadog