The Agent on a Host

Throughout this lab, each section will be broken down into a series of steps. To navigate between sections, click each header to expand or collapse the sections.

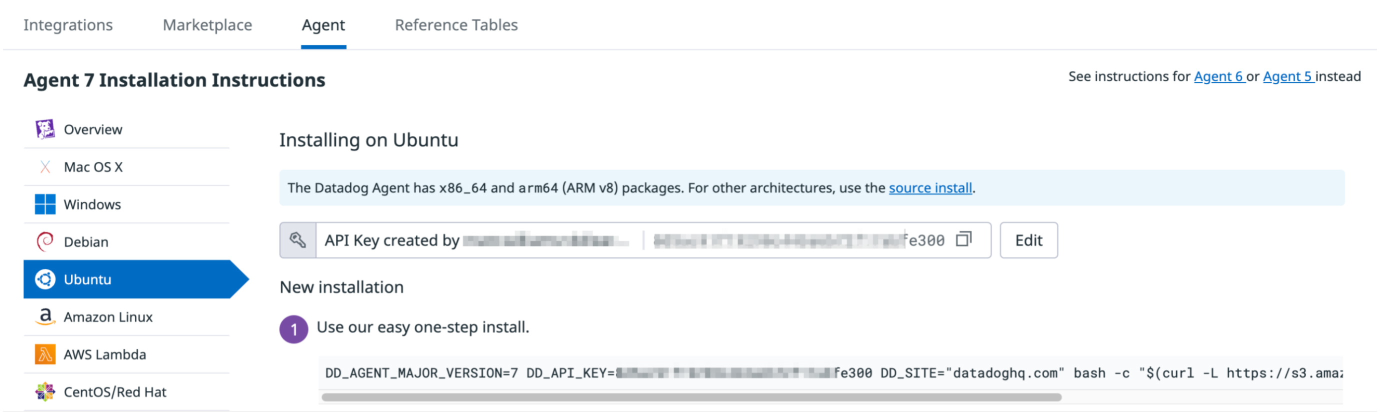
Make sure you are logged into Datadog using the Datadog training account credentials provisioned for you. You can find that information by running `creds` in the lab terminal.

Install the Agent on a Host

In this lab, the host is a virtual machine. The following steps are exactly the same for a physical box.

1. Log in to Datadog using the Datadog training account credentials displayed in the terminal.
2. Navigate to **Integrations** > **Agent** and click on **Ubuntu**.
3. Click the **Select API Key** button and select the API key. The last four digits of the **Key** (not **Key ID**) should match your Datadog training account credentials.

   Select the **Use API Key** button to have your API Key automatically populated in the following **one-step install** command.
4. Copy the full, **one-step install** command to install the Agent.



Agent 7 Installation Instructions

See instructions for Agent 6 or Agent 5 instead

Installing on Ubuntu

The Datadog Agent has x86_64 and arm64 (ARM v8) packages. For other architectures, use the source install.

API Key created by [redacted] [redacted]fe300

New installation

1 Use our easy one-step install.

```
DD_AGENT_MAJOR_VERSION=7 DD_API_KEY=[redacted]fe300 DD_SITE="datadoghq.com" bash -c "$(curl -L https://s3.ama
```

5. Back in the lab terminal, paste the command and press Enter to execute it.

   It should take about a minute to fully install the Agent. When complete, you will see a confirmation message:

```
Your Agent is running and functioning properly. It will continue to run in the
background and submit metrics to Datadog.

If you ever want to stop the Agent, run:

     systemctl stop datadog-agent

And to run it again run:

     systemctl start datadog-agent
```

6. To verify that everything is running the way it should, run the following command:

   `datadog-agent status`

7. Scroll up and review what is shown. Notice that the last characters of your API Key are listed under **API Keys status**.

   Also notice that the Logs Agent is not running. You'll change that shortly.

8. To see all of the resolved configuration values of the running Agent, run the following command:

   `datadog-agent config`

   This output is verbose, and it can be very useful for debugging.

Configure the Agent on a Host

When running on a host, the Datadog Agent is configured by editing YAML files. On Linux systems, these files are located in`/etc/datadog-agent` by default. These files were created when you ran the one-line installer.

> **Note**: In this lab, there is a symbolic link to the Datadog Agent configuration directory at `/root/lab/datadog-agent`. This provides convenient access to the directory in the IDE.

The file that configures every aspect of the Datadog Agent is `datadog.yaml`. It contains hundreds of configuration options and their defaults. In this section, you'll update this file to set an explicit hostname and to enable logging.

Hostname

An important configuration is the hostname of the machine that the Agent is running on. If not explicitly configured, the Agent will use heuristics to tag metrics, logs, and events with a hostname. You can learn more about these heuristics in the Agent documentation.

You'll configure the Agent with an explicit hostname. This will make it easier to isolate the logs, events, and metrics in the Datadog app. You will use the same hostname for all of the labs in this course, and Datadog will recognize them as the same host even though they are unique virtual machines.

1. In the IDE, expand the `datadog-agent` directory.
2. Open `datadog.yaml`.
3. Notice that your API key is already configured on line 11. The one-step Agent installation command you ran included your API key, and the installation process wrote it into this file for you.
4. Scroll down to find the line with `# hostname: <HOSTNAME_NAME>`.

   This configuration option will be ignored by the Agent because it is commented-out with the `#` character.

5. Uncomment this line by removing the `#` character and the space following it. Remember, the IDE will save files for you automatically.
6. Replace `<HOSTNAME_NAME>` with the following value:

   `dd101-sre-host`

That section should look like the following:

```
## @param hostname - string - optional - default: auto-detected
## @env DD_HOSTNAME - string - optional - default: auto-detected
## Force the hostname name.
#
hostname: dd101-sre-host
```

Logging

Logging is not enabled by default, so you will edit `datadog.yaml` to enable logging on this host.

1. In `datadog.yaml`, scroll down to the section titled **Log Collection Configuration** around line 875.

   > **Tip**: You can use the IDE's search function by pressing `Ctrl/Cmd+F`.

2. Find the line with `logs_enabled`.

   Set it to `true` and uncomment the line by removing the leading `#` and space characters. That section should look like the following:

   ```
   ###################################
   ## Log collection Configuration ##
   ###################################

   ## @param logs_enabled - boolean - optional - default: false
   ## @env DD_LOGS_ENABLED - boolean - optional - default: false
   ## Enable Datadog Agent log collection by setting logs_enabled to true.
   #
   logs_enabled: true
   ```

3. Back in the lab terminal, run the following command to restart the Agent:

   ```
   systemctl restart datadog-agent
   ```

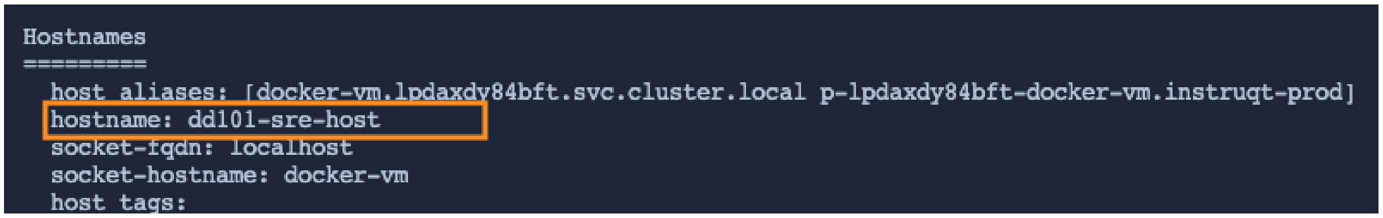
4. Run the following command to verify the status:

   ```
   datadog-agent status
   ```

   > **Note**: If running the `status` command results in an error, you should see the line that the parser got up to. Remember that spacing in a YAML file is very important.

5. Scroll through the status output to find the **Logs Agent** section. Notice that it is now populated.

6. Scroll to the top of the output to find the **Hostnames** section. Here you will see the hostname you explicitly configured:

   ```
   Hostnames
   =========
     host aliases: [docker-vm.lpdaxdy84bft.svc.cluster.local p-lpdaxdy84bft-docker-vm.instruqt-prod]
     hostname: dd101-sre-host
     socket-fqdn: localhost
     socket-hostname: docker-vm
     host tags:
   ```

Also notice the `host_aliases` line. These are other candidate hostnames the Agent discovered. You can see that these are not as human-friendly as the hostname you configured.

Back in the IDE, in `datadog.yaml`, find the settings for APM, which is short for Application Performance Monitoring.

You will see that it is `enabled`, but commented out. This is how Datadog's configuration files document default values. You only need to uncomment things that you want to change. We'll leave this commented for now since we are not changing the value.

Feel free to scroll through this file to get a sense of the configuration options for the Datadog Agent.

Configure Agent Checks

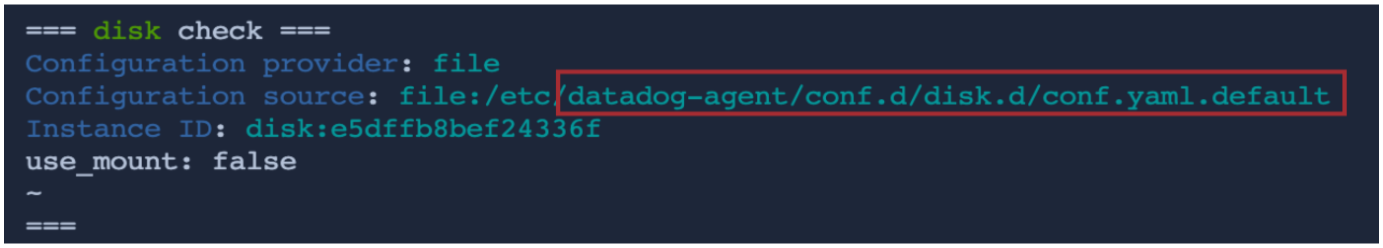
The Datadog Agent runs a core set of checks by default. These include `cpu`, `disk`, `memory`, `uptime`, and more. When you run the `datadog-agent status` command, these appear in the **Collector > Running Checks** section of the output.

Each of these checks has a corresponding configuration file located in a subdirectory of `datadog-agent/conf.d/`.

1. To see a concise list of the checks the Agent is running and their configuration file paths, run the following command in the terminal.

   `datadog-agent configcheck`

   You should see the **Configuration source** for each of the checks. Notice the subdirectory and filename, such as the following for **disk check**:

   ```
   === disk check ===
   Configuration provider: file
   Configuration source: file:/etc/datadog-agent/conf.d/disk.d/conf.yaml.default
   Instance ID: disk:e5dffb8bef24336f
   use_mount: false
   ~
   ===
   ```

2. Back in the IDE, open that file, `datadog-agent/conf.d/disk.d/conf.yaml.default`. Notice that the configuration files for these checks all end in `.default`.

   As in the main Agent configuration file, the default configuration values for the `disk` check are listed in this file. If you want to customize the `disk` check for your system, you would uncomment an option and change its value.

   For example, if you wanted the Agent to check only physical storage devices, you would uncomment `include_all_devices` and set its value to `false`.

   > **Note**: If you update Agent configuration files, you must restart the Agent for those changes to take effect.

All of the checks in `datadog-agent/conf.d/` are known as Agent-based integrations, a subset of the massive Datadog Integrations ecosystem. As such, you can find documentation for all of them in the Integrations documentation. The Integration docs provide a powerful search interface to quickly find what you're looking for. You'll learn more about other integrations later in this course.

If there is a check that you would like the Agent to run that is not provided by a Datadog integration, you can create your own. See the Writing a Custom Agent Check documentation for details.

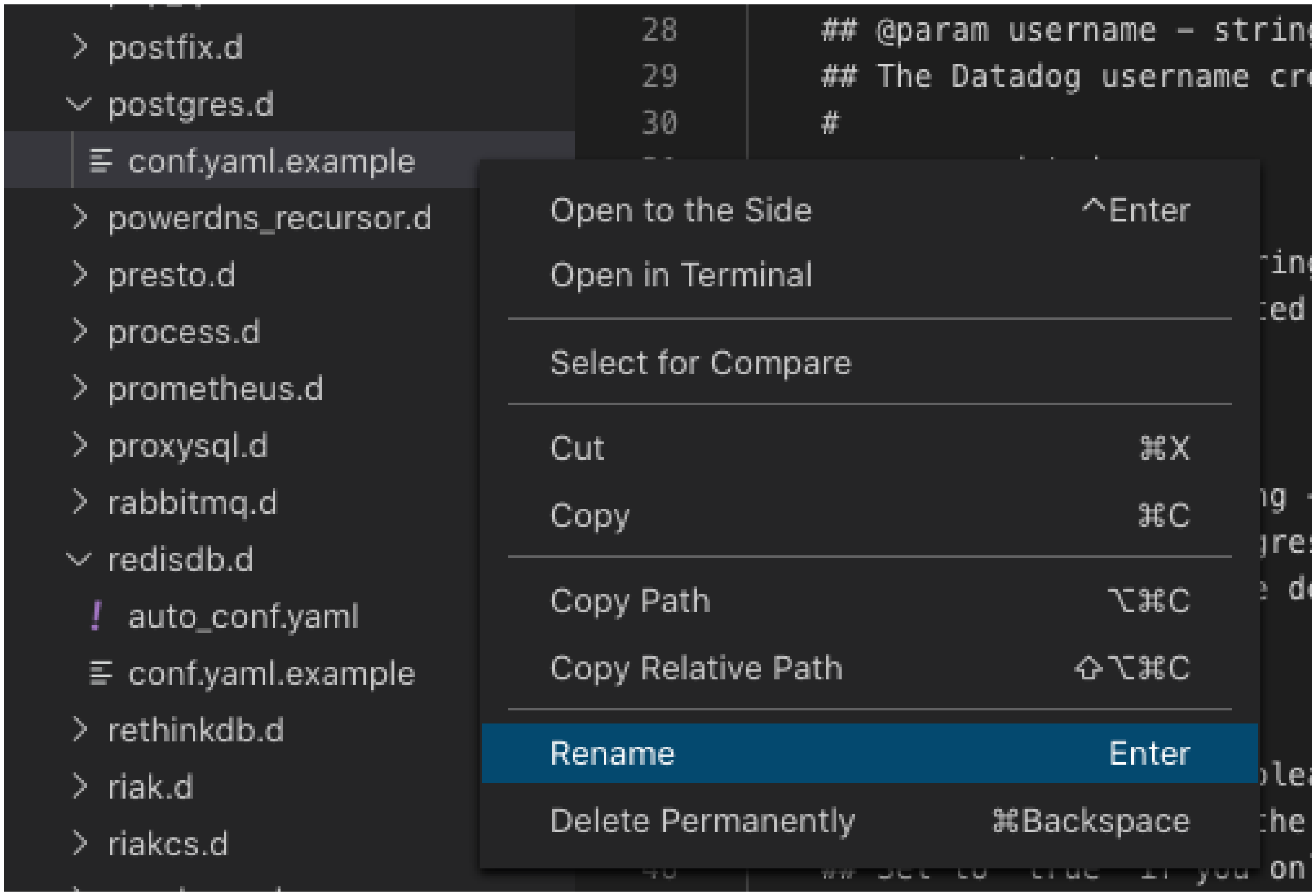
Configure an Integration on a VM

Now you'll learn how to configure the Datadog Agent for a service running on the same host. In this example, you will configure the PostgreSQL Integration to read the logs of a local PostgreSQL server, and to connect to the server to query metadata. You don't need to focus on the specifics of this integration, just the concepts.

1. In the IDE, open `datadog-agent/conf.d/postgres.d/conf.yaml.example`.

   Each Integration configuration directory contains an `.example` file documenting all the configurable parameters. To modify a configuration, you can start by removing the `.example` suffix or copying it to a new file named `conf.yaml` in the same directory.

2. Right-click on `conf.yaml.example` and rename it to `conf.yaml`:

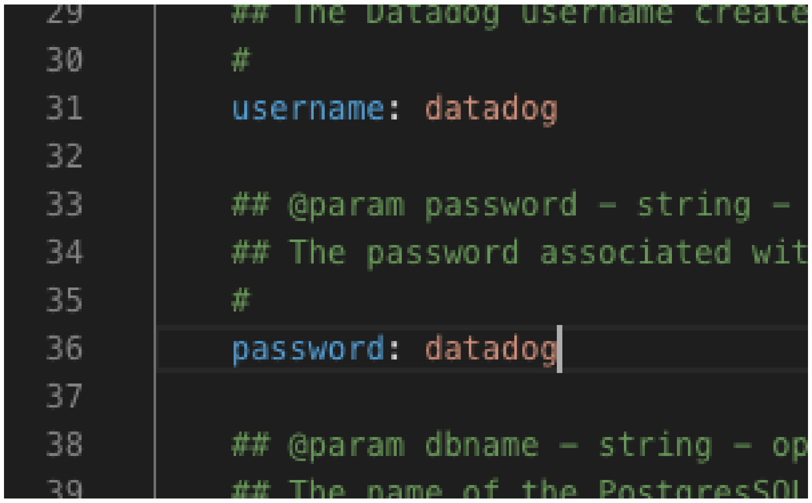


```
28      ## @param username - strin
29      ## The Datadog username cr
30      #
```

3. Around lines 21 to 42 is where you will find the most common configuration parameters for a PostgreSQL database service. You'll see that `host` and `username` are required for the Datadog Agent to connect to the service and query metadata.

4. The `datadog` user has been granted access to the local PostgreSQL server with the password `datadog`.

   Update line 36 to remove the comment and replace `<PASSWORD>` with `datadog`:



```
29      ## The Datadog username create
30      #
31      username: datadog
32
33      ## @param password - string -
34      ## The password associated wit
35      #
36      password: datadog
37
38      ## @param dbname - string - op
39      ## The name of the PostgresSQL
```

   Ensure that key `password` is aligned with `username`, as shown in the above screenshot.

5. Scroll to the bottom of the file to find **Log Section**. This is where you will configure the Agent to collect logs for the PostgreSQL service.

6. Uncomment all the lines in the `logs` block.

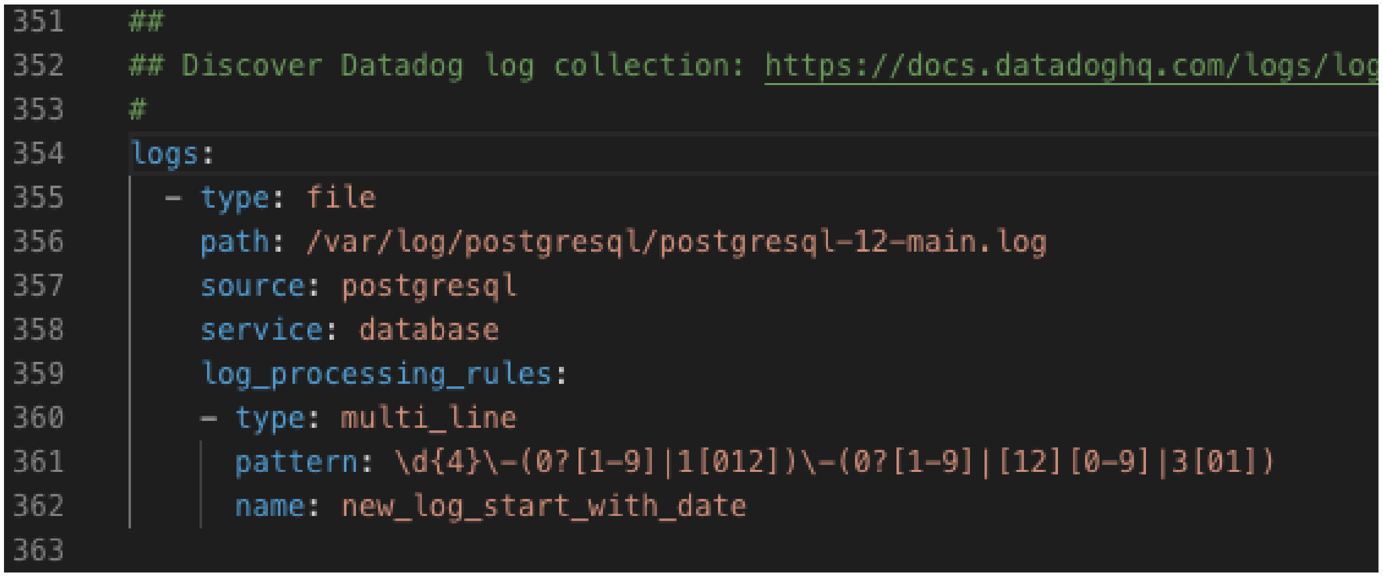
7. Set `path` to the following value:

   `/var/log/postgresql/postgresql-12-main.log`

This is where the PostgreSQL server is writing its logs in the filesystem.

8. Set `service` to `database`. This will add the tag `service:database` to the log entries collected by the Agent.

   Your `logs:` block should look like the following:



```
##
## Discover Datadog log collection: https://docs.datadoghq.com/logs/log
#
logs:
  - type: file
    path: /var/log/postgresql/postgresql-12-main.log
    source: postgresql
    service: database
    log_processing_rules:
    - type: multi_line
      pattern: \d{4}\-(0?[1-9]|1[012])\-(0?[1-9]|[12][0-9]|3[01])
      name: new_log_start_with_date
```

9. By default, PostgreSQL logs are readable only by the `postgres` user and group.

   Back in the lab terminal, add the Datadog Agent user to the `postgres` group by running the following command:

```
usermod -a -G postgres dd-agent
```

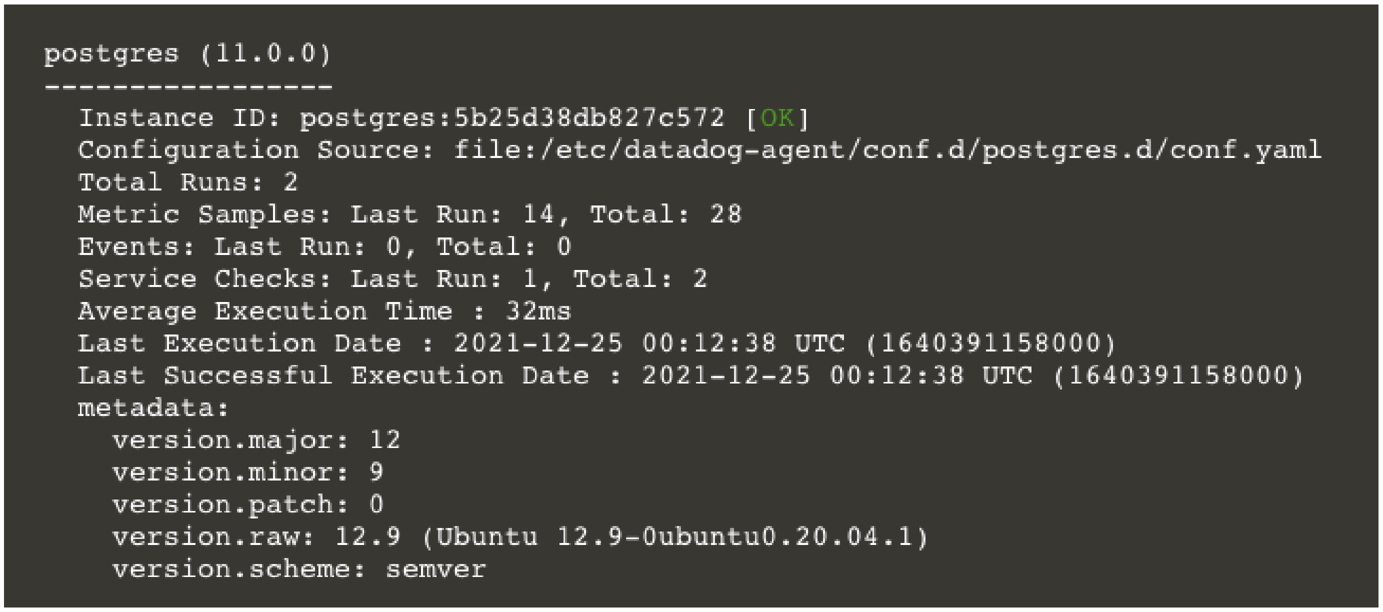
10. Restart the Datadog Agent with by running the following command:

```
systemctl restart datadog-agent
```

11. After a few seconds, run the following command:

```
datadog-agent status
```

12. Scroll up to the **Running Checks** section and find the **postgres** entry:



```
postgres (11.0.0)
-----------------
  Instance ID: postgres:5b25d38db827c572 [OK]
  Configuration Source: file:/etc/datadog-agent/conf.d/postgres.d/conf.yaml
  Total Runs: 2
  Metric Samples: Last Run: 14, Total: 28
  Events: Last Run: 0, Total: 0
  Service Checks: Last Run: 1, Total: 2
  Average Execution Time : 32ms
  Last Execution Date : 2021-12-25 00:12:38 UTC (1640391158000)
  Last Successful Execution Date : 2021-12-25 00:12:38 UTC (1640391158000)
  metadata:
    version.major: 12
    version.minor: 9
    version.patch: 0
    version.raw: 12.9 (Ubuntu 12.9-0ubuntu0.20.04.1)
    version.scheme: semver
```

   > **Note**: If you don't see an entry for **postgres**, run the `status` command after a minute. It can take a little while for the Agent to run checks after restarting.

13. Scroll down to the **Logs Agent** section and find the **postgres** entry:

```
==========
Logs Agent
==========

    Sending compressed logs in HTTPS to agent-http-intake.logs.datadoghq.com on port 443
    BytesSent: 604
    EncodedBytesSent: 343
    LogsProcessed: 3
    LogsSent: 2

  postgres
  --------
    - Type: file
      Path: /var/log/postgresql/postgresql-12-main.log
      Status: OK
      Inputs:
        /var/log/postgresql/postgresql-12-main.log
      BytesRead: 365
      Average Latency (ms): 20
      24h Average Latency (ms): 20
      Peak Latency (ms): 62
      24h Peak Latency (ms): 62
      MultiLine matches: 3
```

You have now configured the Datadog Agent to collect metrics and logs from the PostgreSQL service. These principles are the same for many other integrations. You'll learn more about Datadog Integrations later in the course.

Your Infrastructure in the Datadog App

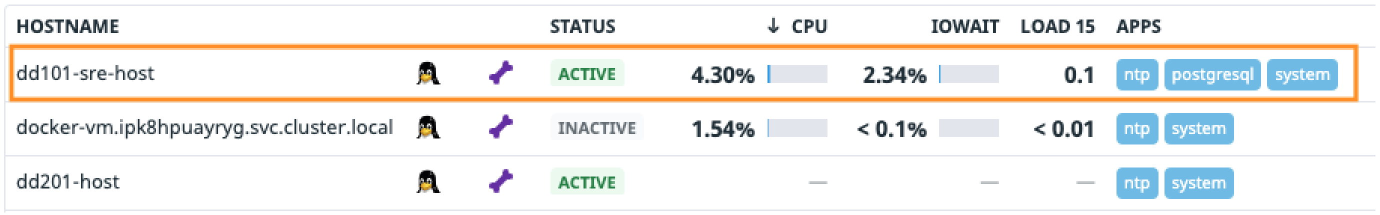
You know that the Agent is sending metrics, events, and logs to Datadog. Take a brief look at the results in the Datadog app:

1. Make sure you are logged into Datadog using your Datadog training account credentials.
2. Navigate to Infrastructure.

   You might see more than one host in this list. Two of them are the same, but with different names–the hostname before setting it in `datadog.yaml`, and the hostname afterward. It will take a little while for Datadog to catch up and deactivate the hold hostname.

   If you see more than two hosts in the infrastructure list, it is likely because you have taken other courses in the past 2 hours. Each course runs its labs in virtual machines using a unique hostname.

   The hostname you'll use in this course is `dd101-sre-host`:



HOSTNAME	STATUS	↓ CPU	IOWAIT	LOAD 15	APPS
dd101-sre-host	ACTIVE	4.30%	2.34%	0.1	ntp postgresql system
docker-vm.ipk8hpuayryg.svc.cluster.local	INACTIVE	1.54%	< 0.1%	< 0.01	ntp system
dd201-host	ACTIVE	—	—	—	ntp system

3. Click on `dd101-sre-host` to open the details side panel. Under **Datadog**, you will see many tags that the Agent automatically added to this host.
4. Navigate to Logs.

   If you see an introductory page asking you to enable logs, click the **Get started** button to close it.

   There are only a few log lines here from the PostgreSQL server. You'll see plenty of logs here in the next lab.

Feel free to click around and get familiar with the Datadog app. You'll visit most of these products and pages in greater detail in the rest of this course.

Lab Conclusion

Congratulations! You know how to install and configure the Datadog Agent on a VM host.

When you're done, enter the following command in the terminal:

```
finish
```

Click the **Check** button in the lower right corner of the lab and wait for the lab to close down before moving on to the next lesson.