Thanks for your interest in the Enterprise Sales Engineer role at Datadog! Please complete the following hiring challenge to move on to the next stage of the interview process. Before getting started, please be sure to read the full set of instructions.

We ask that you record your responses in a Google doc, and then submit that link plus a PDF of the document when you're finished. You can also submit your response as a private gist, if you so choose. While your challenge will have you writing code and creating things in the Datadog UI, the final submission will be the write-up created in the aforementioned Google Doc / gist.

Responses are typically submitted within a week of receiving the challenge, but please let us know if you require more time.

Some tips on content and style

- *We'd rather see you do a few things really well than a lot of things decently. We have over 400 integrations - there's no way you will get to use them all, so instead focus on what interests you and what shows off your strengths.*
- *In general, we are more interested in the process than the results. Being able to articulate technical concepts and business value well is a crucial part of being a Sales Engineer. You may hit some dead ends as you work through this challenge - that's ok! Make sure you document your steps and your troubleshooting approach, even if you run into issues along the way.*
- *The tone you should use in your responses should be as if you were creating a customer facing deliverable (e.g. PoV report). We recommend providing screenshots and code snippets as much as possible. For purposes of anonymity, we ask that you do not include links to any sites that would identify yourself (i.e. your personal Github).*
- *Be creative! Sales Engineers need to be good problem solvers - don't be afraid to think outside the box. There are often multiple ways to solve a problem or accomplish what you are looking to do.*

Initial Setup - Sign up for a Datadog Account

To do this, go to https://app.datadoghq.com/signup and enter your email, full name, etc. For the Company field, please enter “Datadog Recruiting Candidate”.

Part 1 - Create Your Datadog Environment

The purpose of this section is to setup a Datadog Agent on a server and utilize some of the Datadog features to start monitoring your environment.

There are tons of integrations and features that can be used, and you could theoretically spend days getting everything set up. Instead, we want you to focus on a few programs and features that you’re most familiar with. The end goal is to get you familiar with how Datadog works with hands on experience, and to get an understanding of some of your strengths and aptitudes.

While this is a mostly open-ended assignment, we do have a few requirements:

- The agent must be installed on either a VM or as part of a containerized environment.
    - You can spin up a fresh Linux VM via Vagrant or other tools so that you don’t run into any OS or dependency issues.
    - You can utilize a Containerized approach with Docker for Linux and our dockerized Datadog Agent image.
- All work done must be your own.
- When you are done with the assignment, please email us the link to your Google doc, as well as a PDF attachment of the same.

Possible project components (See examples/tips). The possibilities and permutations of projects are innumerable given how many integrations, applications, and systems Datadog can monitor. That being said, we would love to see some combination of the different ‘pillars’ and capabilities of the product for this portion. Successful submissions will often include 2-3 of these items. Please consider the following list a jumping off point and feel free to combine these in whatever way seems most exciting to you:

- Infrastructure Integrations
    - Establish an integration with any technology you are familiar with, and utilize any of the non-default parameters in the yaml file. Feel free to choose more than one and show us how they might interact.
    - Pull in metrics from an AWS/GCP/Azure account
- Logs
    - Configure the Datadog agent to collect logs and create a graph from those logs. Import these into a Timeboard.
- APM

    - Run a web application in any of our supported languages. Use our Trace Search to find something interesting about the traces.
- Synthetics
    - Develop a Datadog Synthetics browser test and build a SLO to track uptime.
- Dashboards
    - Build a custom dashboard, you can use an out of the box dashboard from an integration as a starting point
- Monitors
    - Create a monitor that sends a notification to your email for an alert, and a notification to a Slack channel for a warning.
    - Create a composite monitor. The sub-monitors should include a metric monitor, a host monitor, and an integration monitor.
- Metrics
    - Submit meaningful custom metrics using Dogstatsd and display them on a dashboard
- API
    - Use Datadog's API to submit custom metrics, create dashboards / monitors, query data, or perform some other functionality.
- Containers
    - Install an agent in a Kubernetes environment
    - Utilize Autodiscovery

Part 1 Examples*

*note: these are merely suggested workflows for completing part 1 - they should be treated as inspiration. Your journey may look a lot different, so don't feel obligated to replicate these in any way, both in content and length.

- Example 1:
    - I installed the agent on an AWS EC2 instance and then set up an integration for MongoDB, since I had an app that already used this.
    - I then set up APM for my flask app and then used the app to simulate activity and get some traces in Datadog.

```python
from flask import Flask
import logging
import sys

# Have flask use stdout as the logger
main_logger = logging.getLogger()
main_logger.setLevel(logging.DEBUG)
c = logging.StreamHandler(sys.stdout)
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
c.setFormatter(formatter)
main_logger.addHandler(c)

app = Flask(__name__)

@app.route('/')
def api_entry():
    return 'Entrypoint to the Application'

@app.route('/api/apm')
def apm_endpoint():
    return 'Getting APM Started'

@app.route('/api/trace')
def trace_endpoint():
    return 'Posting Traces'

if __name__ == '__main__':
    app.run(host='0.0.0.0', port='5050')
```

    - Next, I set up a basic dashboard for my MongoDB metrics and a monitor for my Python traces, and played around with the configuration options. I was able to create some interesting Monitor notifications and made a nice looking dashboard.
- Example 2:
    - I deployed my agent on Heroku and traced an existing Rails application there. I also set up the postgres integration on this agent.

    - Using the data from the web app and postgres integration, I created a dashboard and tried to do some correlation on database performance when I made a bunch of queries to it through the app.
    - I then set up a monitor first on some APM metric and then combined it with a postgres metric to make a composite monitor. I added a Webhook notification as part of the monitor.
- Example 3:
    - I spun up Minikube/docker for one of the applications I've built (with postgres and redis/cache) and used this to see where my application bottlenecks/could be optimized in Trace Search.
    - I then built a dashboard on resource consumption of the host/env for the application.
- Example 4:
    - I spun up an instance in GCP (you get a number of free credits) and installed the agent and a few other integrations. I then set up a basic monitor on instance resources and cloned the default GCP dashboard.
    - Next I investigated how Datadog is different than Stackdriver by creating a Dashboard with GCP integration metrics next to basic agent metrics.
- Example 5:
    - I spun up a Linux VM in Vagrant, installed Postgres, and installed the agent. Once I started getting some metrics here, I created a Frankenstein dashboard that pulled in as many widgets as possible. Different graph views, event streams, etc.using system metrics and Postgres metrics.
    - Next I pulled in logs from a Node application I ran locally on my host. Once I had some logs coming in, I added the logs stream to the dashboard I created. I also created a logs monitor to get alerts when too many error logs were created.
- Example 6:
    - I installed the Datadog agent on my Vagrant box and waited for some system metrics to come in. Once that happened, I utilized Terraform to set up a series of dashboards, monitors, and downtimes.
    - I also looked into Chef as a way to configure multiple agents. While I didn't have a large enough environment to necessitate something like this, I documented all the steps I was able to complete.

Part 2 - Questions

Please answer any **TWO** of the following questions. Again, we are more interested in your approach than the final results.

Treat this section as if you were answering/addressing a detractor on a conference call, a tradeshow or a demonstration. What information would you want to get out of them? If the issue or concern isn't obvious, what questions would you ask to gather more information (hint...understand their pain)? Bonus points if you include any Datadog differentiators that provide required capabilities to address their issues / concerns.

Feel free to include any screenshots, links to documentation or blog articles where appropriate.

Objection / Inquiry 1: from prospective customer:
"My team is too busy fighting fires to evaluate another monitoring tool."

- Kate the SRE Manager

Objection / Inquiry 2: from prospective customer:
"We're a mobile gaming company about to release an online multiplayer game that's been super hyped up in the media. We're bracing for a huge influx of traffic on launch day so we'll be scaling up very quickly. Can Datadog help during this critical period?"

- Randy the Product Manager

Objection / Inquiry 3: from prospective customer:
"We're using a hosted ELK stack and it works well enough. Tell me why Datadog's log tool is better than what we have today. How can it help us?"

- Greg the Sr. SRE

Objection / Inquiry 4: from prospective customer:
"I hear Datadog is great at monitoring Public Cloud Infrastructure but we're not moving to the Cloud, we don't need another monitoring solution."

- Pierre the VP of Infra

Objection / Inquiry 5: from prospective customer:
"I'm not sure our security team would let us use a SaaS based monitoring solution. My team tells me we MUST have our infrastructure performance data on premise."

- Hector the Sys Admin