The Agent in a Container

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.

Configure docker-compose.yml

"Installing" the Agent container is done by specifying which Docker image to use. Rather than configuration files, the container Agent relies on environment variables and container labels, which you'll implement in this lab.

1. In the IDE, open `docker-compose.yml`.
2. Get familiar with the structure of this file. There is a block for each Storedog service: `discounts`, `frontend`, `advertisements`, and `db`.

   You can ignore `puppeteer`, which is generating traffic to the Storedog application.
3. To add the Datadog Agent as a service, click on the following block of code to copy it:

```
datadog:
    image: 'datadog/agent:7.31.1'
    environment:
      - DD_API_KEY
      - DD_LOGS_ENABLED=true
      - DD_LOGS_CONFIG_CONTAINER_COLLECT_ALL=true
      - DD_PROCESS_AGENT_ENABLED=true
      - DD_DOCKER_LABELS_AS_TAGS={"my.custom.label.team":"team"}
      - DD_TAGS='env:dd101-dev'
      - DD_HOSTNAME=dd101-dev-host
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - /proc/:/host/proc/:ro
      - /sys/fs/cgroup/:/host/sys/fs/cgroup:ro
```

4. Paste it into `docker-compose.yml` under the comment `# paste agent block here`.

   > **Note:** Make sure the indentation for this service is correct and matches the indentation of other services. The IDE will automatically save the file when you make changes.

Take a look at the details:

- `image: 'datadog/agent:7.31.1` specifies the specific Agent Docker image to use for the container.
- The `environment` block sets the specified environment variables in the Agent container:
  - **DD_API_KEY**: This is required by the Agent to submit metrics and events to Datadog. It's set in the host environment, and you can see it by running `env |grep DD_API_KEY` in the terminal. Because it is not set to a value here, Docker Compose will use the host's environment variable value.
  - **DD_LOGS_ENABLED**: Whether to collect logs.
  - **DD_LOGS_CONFIG_CONTAINER_COLLECT_ALL**: Whether to collect logs emitted by all containers it detects.
  - **DD_PROCESS_AGENT_ENABLED**: Whether to collect live process information.

    - **DD_DOCKER_LABELS_AS_TAGS**: Configures the Agent to treat custom container labels as custom tags. In this case, if the Agent reads the label `my.custom.label.team`, it will assign the value to the tag `team`.
    - **DD_TAGS**: Sets the global `env` tag for all data emitted from the host. In this case, it is setting the special `env` tag to `dd101-dev`.
    - **DD_HOSTNAME**: Explicitly sets the hostname for the Agent.

  Throughout this course, you'll see how valuable tags are in Datadog. For now, focus on how to set them in this environment.

- The `volumes` block mounts the files on the host filesystem into the container. This gives the Agent the ability to query the Docker Daemon for data about the Docker environment, as well as process data from the host itself.

Now that you have added the Agent container to Storedog infrastructure, you can start the application and explore the results.

Start Storedog

In this section, you'll start up the Storedog application and explore its infrastructure and logs in Datadog.

1. In the terminal, ensure you are in the `/root/lab` directory. If not, run this command:

   ```
   cd /root/lab
   ```

2. To start the application stack, run this command:

   ```
   docker-compose up -d
   ```

3. Once the containers are running, run the Agent `status` command:

   ```
   docker-compose exec datadog agent status
   ```

   This command tells `docker-compose` to execute the command `agent status` inside the `datadog` container.

4. Scroll to the **Logs Agent** section of the status output, and notice the **container_collect_all** block.

   > **Note**: If you don't see the **container_collect_all** block under **Logs Agent**, wait a few seconds and run the Agent `status` command again. It needs time to show up.

   Even though you didn't configure the individual services for the Agent, the Agent's `DD_LOGS_CONFIG_CONTAINER_COLLECT_ALL=tr` environment variable tells the Agent to grab all the logs of all the containers running alongside it.

5. Scroll further up to the **Forwarder** section. This is the process that sends everything to Datadog.

   At the bottom of this section, find **API Keys status**. It displays the last few characters of the `DD_API_KEY` environment variable passed in from the host. This should be the same API Key as your Datadog training account credentials.

6. To see the Agent's configuration, run the Agent `config` command:

   ```
   docker-compose exec datadog agent config
   ```

7. Find the `tags` setting. Confirm that `env:dd101-dev` is listed.

Explore the Datadog App

Now that you're confident that the Agent is running according to your configuration, take a look at what the Datadog App is receiving.

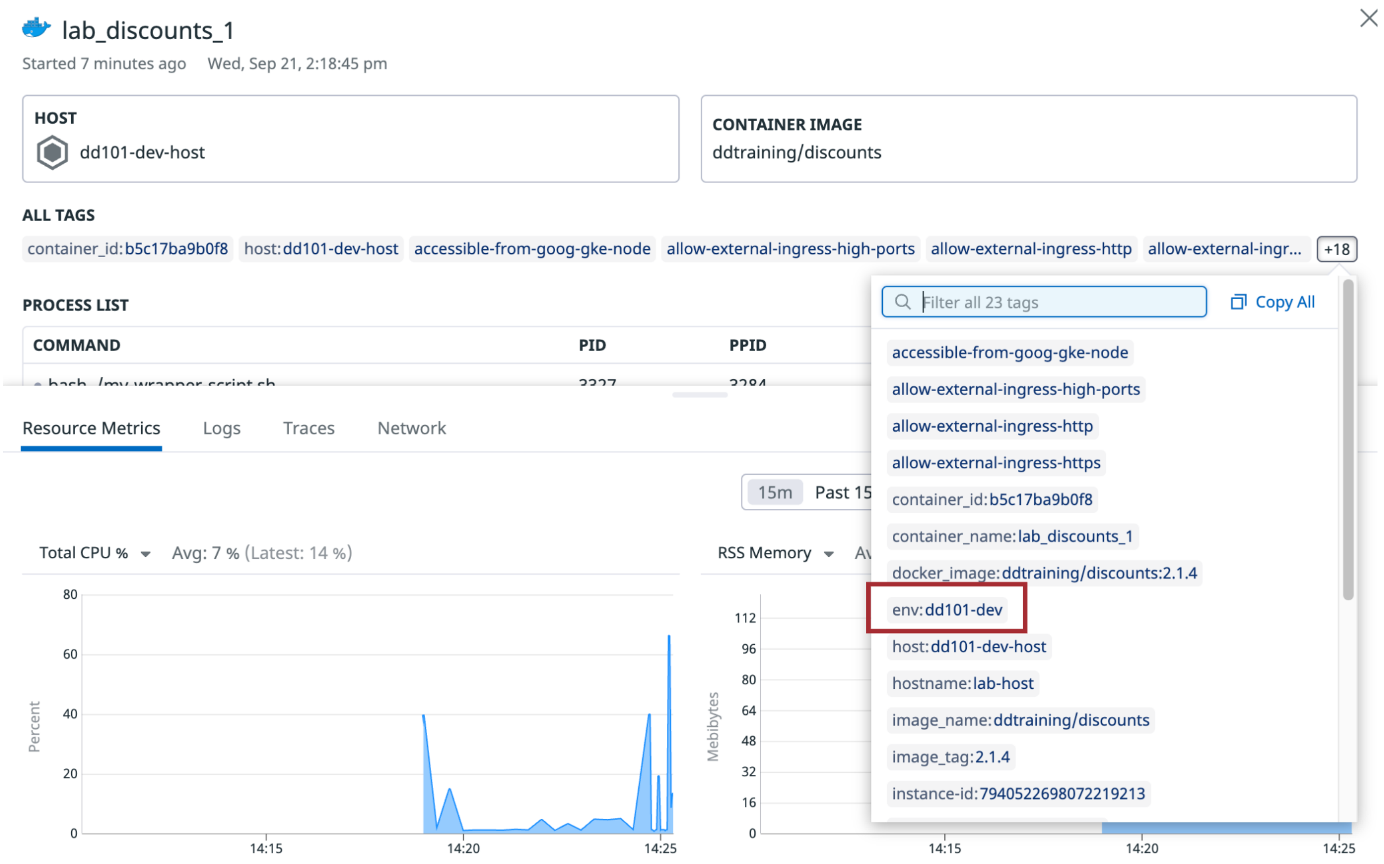
1. Log in to Datadog using the trial credentials the lab created for you. You can run `creds` in the lab terminal whenever you need to retrieve your Datadog training account credentials.

2. Navigate to **Infrastructure**. You will see `dd101-dev-host`, which is the virtual machine hosting your lab environment.

   The Agent container has access to the host that's running the Docker Daemon!

3. Navigate to **Infrastructure > Containers**. Here you can see all of the service containers.

4. Click on `lab_discounts_1` and notice the metadata the Agent captures by default under **ALL TAGS**. You may need to expand the overflow list to see all of the tags.
5. Find the tag `env:dd101-dev`, which was applied to all events, metrics, and logs collected by the Agent.



lab_discounts_1
Started 7 minutes ago Wed, Sep 21, 2:18:45 pm

HOST: dd101-dev-host
CONTAINER IMAGE: ddtraining/discounts

ALL TAGS
container_id:b5c17ba9b0f8 host:dd101-dev-host accessible-from-goog-gke-node allow-external-ingress-high-ports allow-external-ingress-http allow-external-ingr... +18

Filter all 23 tags
accessible-from-goog-gke-node
allow-external-ingress-high-ports
allow-external-ingress-http
allow-external-ingress-https
container_id:b5c17ba9b0f8
container_name:lab_discounts_1
docker_image:ddtraining/discounts:2.1.4
env:dd101-dev
host:dd101-dev-host
hostname:lab-host
image_name:ddtraining/discounts
image_tag:2.1.4
instance-id:7940522698072219213

6. Navigate to **Logs > Search** and see the log entries there.

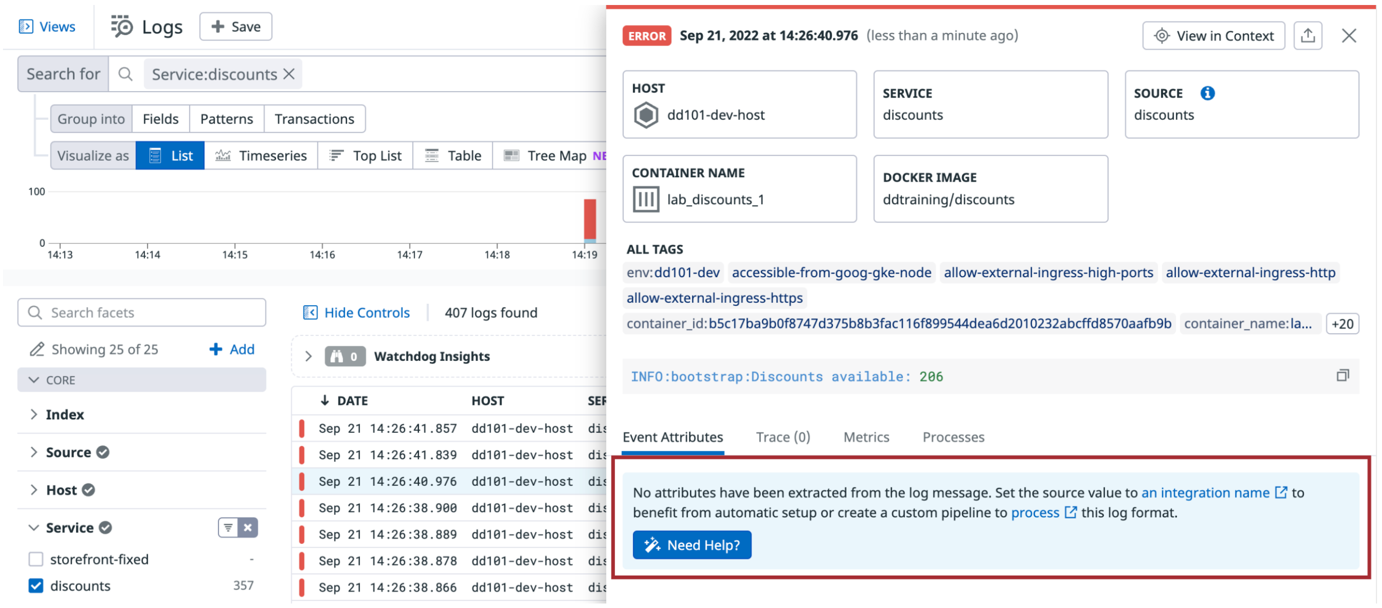
   Note: If you see a warning about logs not being enabled, click the **Get started with Log Management to view container logs** link. This will enable logs for you. Once logs are enabled, go back to view the logs.

   You'll see a lot of log entries here.
7. In the facets panel on the left, expand the **Service** section and select **discounts** to only show logs for the discounts service.

   You are able to see logs from the `discounts` service because the Agent is configured to capture logs from all containers. But notice that they are mostly deemed **ERROR**, as indicated by the red marks.

   Note: These service names are inferred from the containers' short image names. You will override these default service names shortly.
8. Click on a `discounts` log entry. In addition to the log content, the Agent was able to capture a lot of metadata, such as the **CONTAINER NAME**, **DOCKER IMAGE**, **SERVICE** and **SOURCE** based on the service name.

   Notice under **Event Attributes**, there is a message saying that you should set the source to an integration name.



ERROR Sep 21, 2022 at 14:26:40.976 (less than a minute ago)

HOST dd101-dev-host	SERVICE discounts	SOURCE discounts
CONTAINER NAME lab_discounts_1	DOCKER IMAGE ddtraining/discounts	

ALL TAGS
env:dd101-dev accessible-from-goog-gke-node allow-external-ingress-high-ports allow-external-ingress-http allow-external-ingress-https container_id:b5c17ba9b0f8747d375b8b3fac116f899544dea6d2010232abcffd8570aafb9b container_name:la... +20

```
INFO:bootstrap:Discounts available: 206
```

No attributes have been extracted from the log message. Set the source value to an integration name to benefit from automatic setup or create a custom pipeline to process this log format.

DATE	HOST	SER
Sep 21 14:26:41.857	dd101-dev-host	dis
Sep 21 14:26:41.839	dd101-dev-host	dis
Sep 21 14:26:40.976	dd101-dev-host	dis
Sep 21 14:26:38.900	dd101-dev-host	dis
Sep 21 14:26:38.889	dd101-dev-host	dis
Sep 21 14:26:38.878	dd101-dev-host	dis
Sep 21 14:26:38.866	dd101-dev-host	dis

You'll see how to do this later in the course.

Now that the Agent is configured, you will help it tag the other service containers by configuring their labels.

Configure the Discount Service

Next, you'll add labels to the the discounts service so the Agent will know how tag individual services. Specifically, you set the tags `env`, `service`, and `version` which enable Unified Service Tagging. You'll learn more about this when you work with APM and logs later in the course.

1. In the IDE, open `docker-compose.yml`.
2. Add the following labels to the `discounts` service under the comment `# paste discounts labels here`:

   ```
   labels:
         com.datadoghq.tags.env: 'dd101-dev'
         com.datadoghq.tags.service: 'discounts-service'
         com.datadoghq.tags.version: '2.1'
         my.custom.label.team: 'discounts'
   ```

   `com.datadoghq.tags.env: 'dd101-dev'` is technically unnecessary in this case because the Agent will set that by default for all services. But there are many use cases where a service would have a different `env` value than the Agent's default.

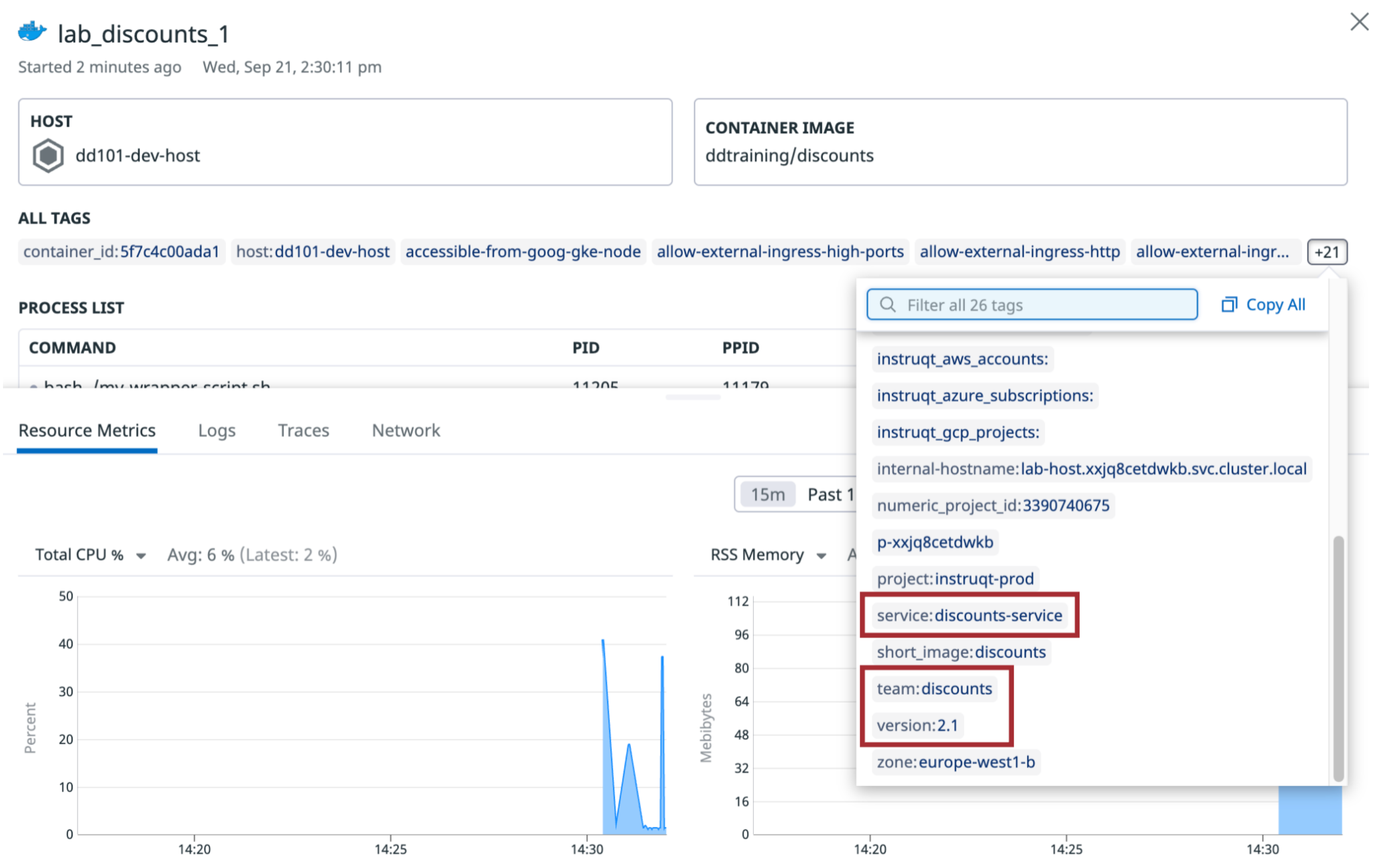
   The last label sets the custom tag `team:discounts`, as declared by `DD_DOCKER_LABELS_AS_TAGS` in the `agent` service.

3. In the terminal, restart the application stack by running the following command:

   ```
   docker-compose down && docker-compose up -d
   ```

   Wait for Docker Compose to complete before proceeding.

4. In Datadog, navigate to **Infrastructure > Containers**. Wait for the new containers to show up. The **STARTED (AGO)** column should show that the containers started 1 minute ago.
5. Click on **lab_discounts_1**. Notice the new `service`, `team` and `version` tags added under **ALL TAGS**. You may need to expand the overflow list to see all of the tags:



6. Navigate to **Logs**.
7. In the facets panel on the left, filter by the `discounts` service, and click on a log entry.

   Again, you'll see that new tags you added, including `service:discounts-service`. The default `service:discounts` tag from the application will persist until you override it with APM later in the course.

Now that the discounts service container is labeled and tagged correctly, you can configure the remaining services.

Configure the Remaining Services

Now that you have given the discounts service unique tags, you can configure the rest of the application services.

For your convenience, there is a fully configured Docker Compose file already in the filesystem.

1. In the terminal, stop the application services by running the following comand:

   ```
   docker-compose down
   ```

2. Replace the existing `docker-compose.yml` with the fully configured Docker Compose file by running the following command:

   ```
   cp /root/docker-compose-complete.yml /root/lab/docker-compose.yml
   ```

3. Restart the application services again by running the following command:

   ```
   docker-compose up -d
   ```

4. While you wait for that to start up and send data to Datadog, open the new `docker-compose.yml` file in the IDE. If the file was already open, you may need to refresh it or reopen the file.

   Notice that all of the services are now configured using labels similar to the `discounts` service.

   Each service now has `datadog` listed under `depends-on`. This prevents services from starting up before the Agent can gather their data.

5. Navigate back to Datadog to check out the logs and tags for the services.

Lab Conclusion

Congratulations! You know how to install and configure the Datadog Agent container. You learned how to configure the Agent with environment variables, and how to use labels to uniquely tag each container that the Agent observes. You also observed the label values represented as tags in the Datadog App, in both the container details and log entries.

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.