Logs Forwarder
LFB-LOGS - LogForwarded Lambda Created -
https://us-gov-west-1.console.amazonaws-us-gov.com/lambda/home?region=us-gov-west-1#/functions/datadog-forwarder-Forwarder-feCWfv1R5k6N?tab=configure

Using link:-
CF -
https://us-gov-west-1.console.amazonaws-us-gov.com/cloudformation/home#/stacks/create/review?stackName=datad[…]azonaws.com/aws/forwarder/latest.yaml

Windows Event:-
Installation Agent
https://app.datadoghq.com/account/settings/agent/latest?platform=windows

Enable Logs:-
https://docs.datadoghq.com/agent/configuration/agent-configuration-files/?tab=agentv6v7#agent-configuration-directory

`%ProgramData%\Datadog\datadog.yaml`
`logs_enabled: true` in your `datadog.yaml` file.

Then check the status here:-
& "$env:ProgramFiles\Datadog\Datadog Agent\bin\agent.exe" launch-gui
http://127.0.0.1:5002/

Do changes to windows event configuration file:-
https://docs.datadoghq.com/integrations/win32_event_log/?tab=logs

`%ProgramData%\Datadog\conf.d\win32_event_log.d/conf.yaml`

Change conf.yaml.example to conf.yaml

```
Unset
# For Logs

logs:
  - type: windows_event
    channel_path: Security
    source: windows.events
    service: Windows_Security
```

```
  - type: windows_event
    channel_path: System
    source: windows.events
    service: Windows_System

  - type: windows_event
    channel_path: Application
    source: windows.events
    service: Windows_Application

  - type: windows_event
    channel_path: Setup
    source: windows.events
    service: Windows_Setup

  - type: windows_event
    channel_path: Microsoft-Windows-SystemDataArchiver/Diagnostic
    source: windows.events
    service: Windows_Diagnostic

  - type: windows_event
    channel_path: Microsoft-Windows-TaskScheduler/Operational
    source: windows.events
    service: Windows_TaskScheduler/Operational

  - type: windows_event
    channel_path: Microsoft-Windows-Ntfs/Operational
    source: windows.events
    service: Windows_Ntfs/Operational
```

FOR EVENTS

```
init_config:
    legacy_mode: false
instances:
```

```
- # Event Log API
  path: Security
  service: Security
  filters: {}

- path: Application
  service: Application
  filters: {}

- path: System
  service: System
  filters: {}

- path: Setup
  service: Setup
  filters: {}

- path: Microsoft-Windows-SystemDataArchiver/Diagnostic
  service: Diagnostic
  filters: {}

- path: Microsoft-Windows-TaskScheduler/Operational
  service: Operational-TS
  filters: {}

- path: Microsoft-Windows-Ntfs/Operational
  service: Operational-NTFS
  filters: {}
```

ECS:-

https://docs.datadoghq.com/containers/amazon_ecs/?tab=webui

Task Definition Json file:-
https://docs.datadoghq.com/resources/json/datadog-agent-ecs-logs.json

ECS Log Collection:-
https://docs.datadoghq.com/containers/amazon_ecs/logs/?tab=linux

Process & Network Monitor
https://docs.datadoghq.com/containers/amazon_ecs/?tab=webui

Final

APM

Dummy app node.js -
https://aws.plainenglish.io/deploying-a-node-js-application-container-on-amazon-ecs-e2730d26893f

```
{
   "containerDefinitions": [
       {
           "name": "datadog-agent",
           "image": "public.ecr.aws/datadog/agent:latest",
           "cpu": 100,
           "memory": 512,
           "portMappings": [
               {
                 "hostPort": 8126,
                 "protocol": "tcp",
                 "containerPort": 8126
               }
           ],
           "essential": true,
           "environment": [
               {
                   "name": "DD_LOGS_ENABLED",
                   "value": "true"
               },
               {
```

```
                "name": "DD_API_KEY",
                "value": ""
            },
            {
                "name": "DD_SITE",
                "value": "datadoghq.com"
            },
            {
                "name": "DD_LOGS_CONFIG_CONTAINER_COLLECT_ALL",
                "value": "true"
            },
            {
                "name": "DD_PROCESS_AGENT_ENABLED",
                "value": "true"
            },
            {
                "name": "DD_SYSTEM_PROBE_NETWORK_ENABLED",
                "value": "true"
            }
        ],
        "linuxParameters": {
            "capabilities": {
              "add": [
                "SYS_ADMIN",
                "SYS_RESOURCE",
                "SYS_PTRACE",
                "NET_ADMIN",
                "NET_BROADCAST",
                "NET_RAW",
                "IPC_LOCK",
                "CHOWN"
              ]
            }
        },
        "mountPoints": [
            {
                "sourceVolume": "docker_sock",
                "containerPath": "/var/run/docker.sock",
                "readOnly": true
            },
```

```
            {
                "sourceVolume": "cgroup",
                "containerPath": "/host/sys/fs/cgroup",
                "readOnly": true
            },
            {
                "containerPath": "/sys/kernel/debug",
                "sourceVolume": "debug"
            },
            {
                "sourceVolume": "proc",
                "containerPath": "/host/proc",
                "readOnly": true
            },
            {
                "containerPath": "/opt/datadog-agent/run",
                "sourceVolume": "pointdir",
                "readOnly": false
            },
            {
                "containerPath": "/var/lib/docker/containers",
                "sourceVolume": "containers_root",
                "readOnly": true
            }
        ]
    }
],
"family": "datadog-agent-task",
"volumes": [
    {
        "name": "docker_sock",
        "host": {
            "sourcePath": "/var/run/docker.sock"
        }
    },
    {
        "name": "proc",
        "host": {
            "sourcePath": "/proc/"
        }
```

```
        },
        {
            "host": {
              "sourcePath": "/opt/datadog-agent/run"
            },
            "name": "pointdir"
        },
        {
            "name": "cgroup",
            "host": {
                "sourcePath": "/sys/fs/cgroup/"
            }
        },
        {
            "host": {
              "sourcePath": "/sys/kernel/debug"
            },
            "name": "debug"
        },
        {
            "host": {
              "sourcePath": "/var/lib/docker/containers/"
            },
            "name": "containers_root"
        }
    ]
}
```

Reference:-
https://docs.datadoghq.com/tracing/trace_collection/automatic_instrumentation/dd_libraries/nodejs/

Then we need to install the dd-trace dependencies along with our node.js application's.
So from the server where we are compiling and creating Dockerfile, we need to install the dd-trace dependencies using command:-

Unset

```
npm install dd-trace --save
```

Then we have to combine all the package-lock.json dependencies of application and dd-trace Also update package.json file as well adding dd-trace dependencies:-

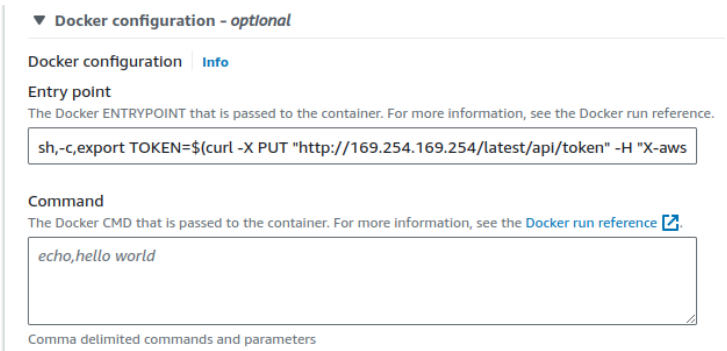
We also need to keep in mind that we need to use the same base image version of **node** in our dockerfile so that it can handle dependencies. To update dependencies we can use this command in the working folder:-npm update --save

```
[ec2-user@ip-172-31-17-175 Nodejs-hello-world]$ cat package.json
{
  "name": "deploying-a-node.js-web-application-on-amazon-ec2",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "jest ./tests/ --forceExit",
    "start": "node index.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "dd-trace": "^5.11.0",
    "dotenv": "^16.4.5",
    "express": "^4.19.2"
  },
  "devDependencies": {
    "axios": "^1.6.8",
    "jest": "^29.7.0",
    "supertest": "^6.3.4"
  }
}
[ec2-user@ip-172-31-17-175 Nodejs-hello-world]$
```

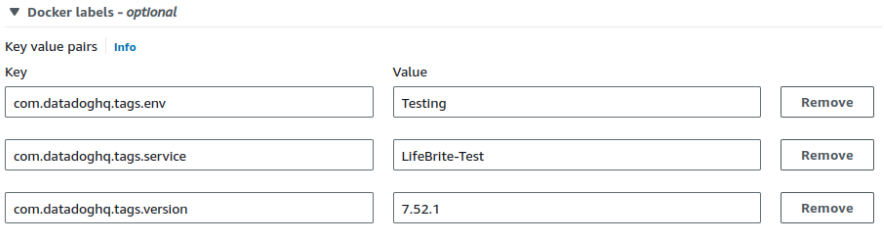
Then in our .js code we need to add this line at the top:-

```javascript
const tracer = require('dd-trace').init();
```

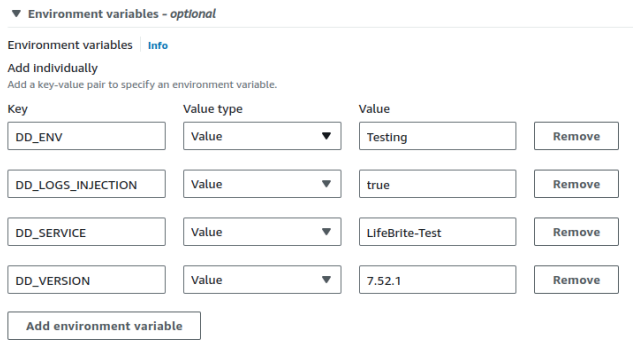
The we have to add entrypoint in task definition
"**entryPoint**": [
"sh",
"-c",
"export TOKEN=$(curl -X PUT \"http://169.254.169.254/latest/api/token\" -H \"X-aws-ec2-metadata-token-ttl-seconds: 21600\"); export DD_AGENT_HOST=$(curl -H \"X-aws-ec2-metadata-token: $TOKEN\" http://169.254.169.254/latest/meta-data/local-ipv4); <Startup Command>"
]



Add **docker labels** as :-



Environment Variables as well:-



Final Json File for Task Definition:-


```
{
   "family": "nodejs",
   "containerDefinitions": [
      {
         "name": "node-hello-world",
         "image": "public.ecr.aws/m4r1s5r9/node-hello-world:latest",
         "cpu": 0,
         "portMappings": [
            {
               "name": "node-hello-world-3000-tcp",
               "containerPort": 3000,
               "hostPort": 3000,
               "protocol": "tcp",
               "appProtocol": "http"
            }
         ],
         "essential": true,
         "entryPoint": [
            "sh",
            "-c",
            "export TOKEN=$(curl -X PUT \"http://169.254.169.254/latest/api/token\" -H \"X-aws-ec2-metadata-token-ttl-seconds: 21600\"); export DD_AGENT_HOST=$(curl -H \"X-aws-ec2-metadata-token: $TOKEN\" http://169.254.169.254/latest/meta-data/local-ipv4); npm start"
         ],
```

```
        "environment": [
            {
                "name": "DD_SERVICE",
                "value": "LB-Test"
            },
            {
                "name": "DD_LOGS_INJECTION",
                "value": "true"
            },
            {
                "name": "DD_ENV",
                "value": "Testing"
            },
            {
                "name": "DD_APPSEC_ENABLED",
                "value": "true"
            },
            {
                "name": "DD_VERSION",
                "value": "7.52.1"
            }
        ],
        "mountPoints": [],
        "volumesFrom": [],
        "dockerLabels": {
            "com.datadoghq.tags.env": "Testing",
            "com.datadoghq.tags.service": "LB-Test",
            "com.datadoghq.tags.version": "7.52.1"
        },
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-create-group": "true",
                "awslogs-group": "/ecs/nodejs",
                "awslogs-region": "us-east-1",
                "awslogs-stream-prefix": "ecs"
            },
            "secretOptions": []
        },
        "systemControls": []
    }
],
"requiresCompatibilities": [
    "EC2"
```

```
    ],
    "cpu": "205",
    "memory": "205",
    "runtimePlatform": {
        "cpuArchitecture": "X86_64",
        "operatingSystemFamily": "LINUX"
    }
}
```

To enable ASM

https://docs.datadoghq.com/security/application_security/enabling/tracing_libraries/threat_detection/nodejs/?tab=amazonecs

Just need to add Environment variable in our application containers after enabling ASM:-

```
"environment": [
  ...,
  {
    "name": "DD_APPSEC_ENABLED",
    "value": "true"
  }
]
```

https://github.com/riboseinc/terraform-aws-ecs-datadog/blob/master/ecs_datadog.tf