DDA Standalone Application Setup and Deployment Guide

The documentation outlines the steps to:

- Train and compile models using Amazon SageMaker
- Package models that consumed by DDA edge application
- Set up IoT devices and bootstrap them with required dependencies
- Deploy the DDA edge application on the devices, which uses the Triton Inference Server
- How to rebuild DDA edge application with code changes against Greengrass

The key steps include creating the training job, compiling the model, packaging it, and then deploying the application components to the IoT devices using Greengrass V2 directly in customer account.

Supported Device Type

These are the 3 device types we support in DDA edge application running standalone:

1. X86_64 CPU
2. Jetson Xavier JetPack4
3. ARM64 CPU

SageMaker Training and SageMaker Compilation Job

###Section for SageMaker training job and SageMaker compilation job
Customer would use SageMaker algorithm to create training job with LFV models. In the provided notebook, the process is using SageMaker algorithm to create training job on classification or segmentation model.

SageMaker training and compilation Jupyter Notebook(Notebook kernel - conda_python3): SageMaker Training for DDA.ipynb
(Note: The role used to run notebook, should have s3 full access, SageMaker full access and IAM role creation access)

Classification model manifest file in the notebook: train_class (1).manifest
Segmentation model manifest file in the notebook: train_segmentation (1).manifest
To create manifest file, use script: https://github.com/aws-samples/amazon-lookout-for-vision/blob/main/computer-vision-defect-detection/cookie-dataset/getting_started.py

Compilation Job Configuration Map:

Device Type	TargetPlatform OS	TargetPlatform Arch	TargetPlatform Accelerator	Compiler Options
X86_64 CPU	LINUX	X86_64		
Jetson Xavier JetPack4	LINUX	ARM64	NVIDIA	{"cuda-ver": "10.2","gpu-code": "sm_72","trt-ver": "8.2.1"}
ARM64 CPU	LINUX	ARM64		

Model Packaging

###Section for model packaging, package model from compiled artifacts with manifest file. Then create GG component based on that
Pre-requisites for the model packaging:

1. Compressed model artifacts, model pt files etc.
2. Compiled/Compressed model artifacts from above step

Model Packaging in the notebook: gg_component_orchestrator.ipynb

Note!!: Make sure the role associated with the Jupyter notebook has S3 and Greengrass permissions to create the component
How to create a Jupyter notebook in Sagemaker: <see here>
How to upload the notebook in Sagemaker: <see here>

Device Setup

###Section for set up IoT device and bootstrap device

- edge manager config - `edge_manager_agent_config.json`

```
{
  "sagemaker_edge_core_capture_data_batch_size": 10,
  "sagemaker_edge_core_capture_data_buffer_size": 30,
  "sagemaker_edge_core_capture_data_destination": "Disk",
  "sagemaker_edge_core_capture_data_disk_path": "{}/em_agent/capture_data",
  "sagemaker_edge_core_device_name": "{}",
  "sagemaker_edge_core_device_fleet_name": "dda_fleet",
  "sagemaker_edge_core_folder_prefix": "dda_eminfer_l4v_emdatacapture/",
  "sagemaker_edge_provider_provider_path": "{}/packages/artifacts-unarchived/aws.edgem
  "sagemaker_edge_local_data_root_path": "/aws_dda/greengrass/v2/em_agent/local_data",
  "sagemaker_edge_log_verbose": true,
  "sagemaker_edge_core_root_certs_path": "{}",
  "sagemaker_edge_provider_provider": "Aws",
  "sagemaker_edge_provider_s3_bucket_name": "sm_edge_manager_dummy_s3_bucket_not_used_
  "sagemaker_edge_core_region": "{}"
}
```

- bootstrap script
  - Download bootstrap script on device
    - Put file `edge_manager_agent_config.json` in the same folder with bootstrap script.
    - installGreengrassCore.sh
    - **Note: Update the above script with the appropriate IoT thing name/group and the desired AWS region for creating the station(on line 180).**
  - Have AWS creds configured, then run as root user on device
    - This script will install required dependencies to run DDA edge application, create IoT device to manage model components and application
    - Note: If device already has greengrass running, uninstall previous greengrass first. GreenGrass has constraints to run more than one greengrass core service.
  - `chmod +x ./installGreengrassCore.sh`

- `sudo ./installGreengrassCore.sh`

GreenGrass Deployment

###GG Deployment recipe with model component, local server

- Deployment recipe
  - Create a deployment.json file on device
  - Run following command to deploy
  - `ubuntu@ip-10-0-42-25:~$ aws greengrassv2 create-deployment --cli-input-json file://deployment.json`

```
## deployment.json
{
    ## IoT thing arn could be found on IoT core console
    "targetArn": "<<REPLACE_WITH_IOT_THING_ARN_CREATED_ON_LAST_STEP>>",
    "deploymentName": "DDA Demo",
    "components": {
        "aws.greengrass.Cli": {
            "componentVersion": "2.12.0"
        },
        "aws.greengrass.Nucleus": {
            "componentVersion": "2.12.0",
            "configurationUpdate": {
                "merge": "{\"interpolateComponentConfiguration\": true}"
            }
        },
        "aws.greengrass.LocalDebugConsole": {
            "componentVersion": "2.4.1"
        },
        "aws.edgeml.dda.LocalServer": {
            ## Change to local server component version created in the account
            "componentVersion": "1.0.99",
            "configurationUpdate": {
                ## Change station name if needed
                "merge": "{\"StationName\":\"dev-test\"}"
            }
        },
        "aws.greengrass.ShadowManager": {
            "componentVersion": "2.3.5"
        },
        "aws.greengrass.TokenExchangeService": {
            "componentVersion": "2.0.3"
        },
        ## Replace with model component name created from step "Model Packaging"
        "<<model-xxxx>>": {
            "componentVersion": "y.0.0"
        }
    }
}
```

Note: Once the deployment process has completed, please reboot the device to ensure the changes are properly implemented.

Local Server Dev Build

###After GG deployment, local server application should be up and running

Local Server code package: https://code.amazon.com/packages/EdgeMLDefectDetectionLocalServer/trees/mainline

- How to build local server application from GDK after code change
  - Install GDK by following instructions - https://docs.aws.amazon.com/greengrass/v2/developerguide/install-greengrass-development-kit-cli.html
  - Install Docker and Docker-compose
    - Docker-compose is needed for docker image build: https://docs.docker.com/compose/install/standalone/, `docker-compose` is different from `docker compose`.
  - To build the component - `gdk component build`
    - **NOTE!!: gdk component build on arm target can only apply on arm devices, gdk component build on x86 target can only apply on x86 devices. Each build will have a different component version even when built for arm and x86**
    - Build on x86 device → Get local server component version with x86 support
      - AMI id - `ami-0f5a21dcf12e8af39` (us-west-2)
      - AMI name(search for name if not in us-west-2) - `ubuntu-pro-server/images/hvm-ssd/ubuntu-bionic-18.04-amd64-pro-server-20240410`
    - Build on arm device → Get local server component version with arm support
      - AMI id - `ami-0cde78da36f9220f8` (us-west-2)
      - AMI name(search for name if not in us-west-2) - `ubuntu-pro-server/images/hvm-ssd/ubuntu-bionic-18.04-arm64-pro-server-20240410`
    - To have a component version with both platform support, create a new version manually and point to s3 artifact link created by previous versions.
  - To publish the component
    - Export credentials of the AWS account in which you want to publish the component
    - Update `gdk-config.json` to set correct region
    - `gdk component publish`
  - Once new component version created, update deployment recipe with new component version to reflect changes.

Troubleshoot

Potential issues may arise:

1. During device bootstrap, if fails with "The role with name GreengrassV2TokenExchangeRole cannot be found".
   a. Rerun the command to create IoT thing

Appendix

1. Supported library version list: Sheet1: Defect App (DDA) 3p libraries attribution
2. Customer Jira ticket queue tracker: JIRA: DDA - JIRA - Primary Tracker
3. Backlog SIM folder: https://issues.amazon.com/issues/search?q=status%3A(Open)+containingFolder%3A(1a623f3d-57f8-40bb-b88f-b64ebb5f0d2c)&sort=lastUpdatedConversationDate+desc&selectedDocument=c89da2e5-579c-455a-a1bb-9552acb5a72d
4. LFV training with SageMaker algorithm notebook and instruction: https://github.com/aws-samples/amazon-lookout-for-vision/tree/main/computer-vision-defect-detection