Advantech EI-52 installation guide to Edge Insights for Vision

This page will guide you through steps to configure Edge Insights for Vision on ESH, install on target device, pull container from docker hub, and run benchmark application with AI models.

Step 1 : Prepare for target OS system

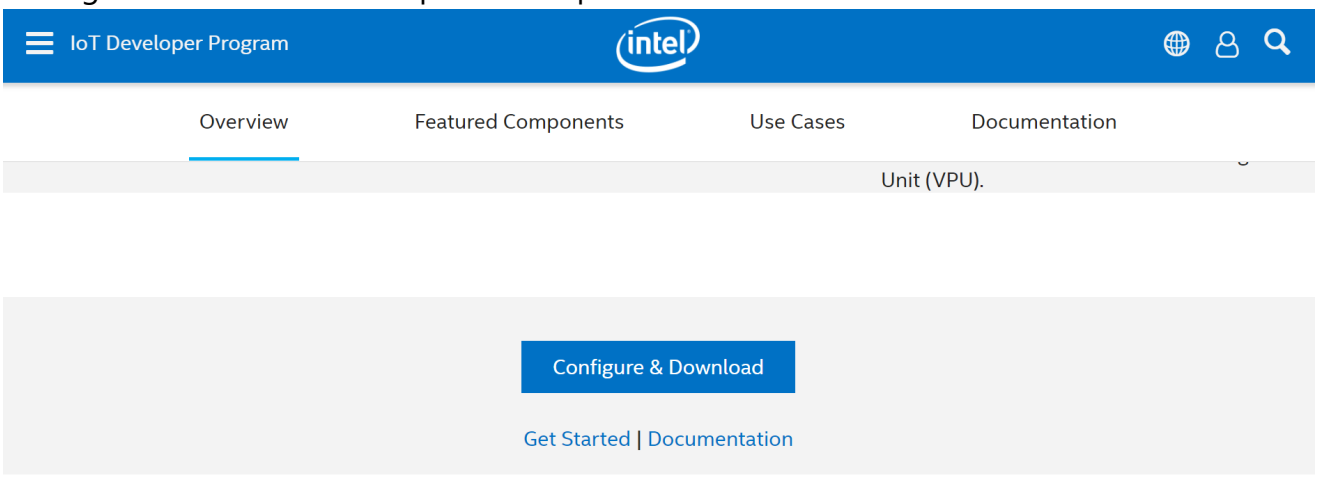
Make sure your target system has a fresh installation of Ubuntu that corresponds to the version of Edge Insights for Vision that you downloaded.
Here I use Ubuntu20.04 LTS as example.

*URL to download Ubuntu 20.04 LTS ISO image:
https://ubuntu.com/download/desktop/thank-you?version=20.04.2.0&architecture=amd64 (https://ubuntu.com/download/desktop/thank-you?version=20.04.2.0&architecture=amd64)

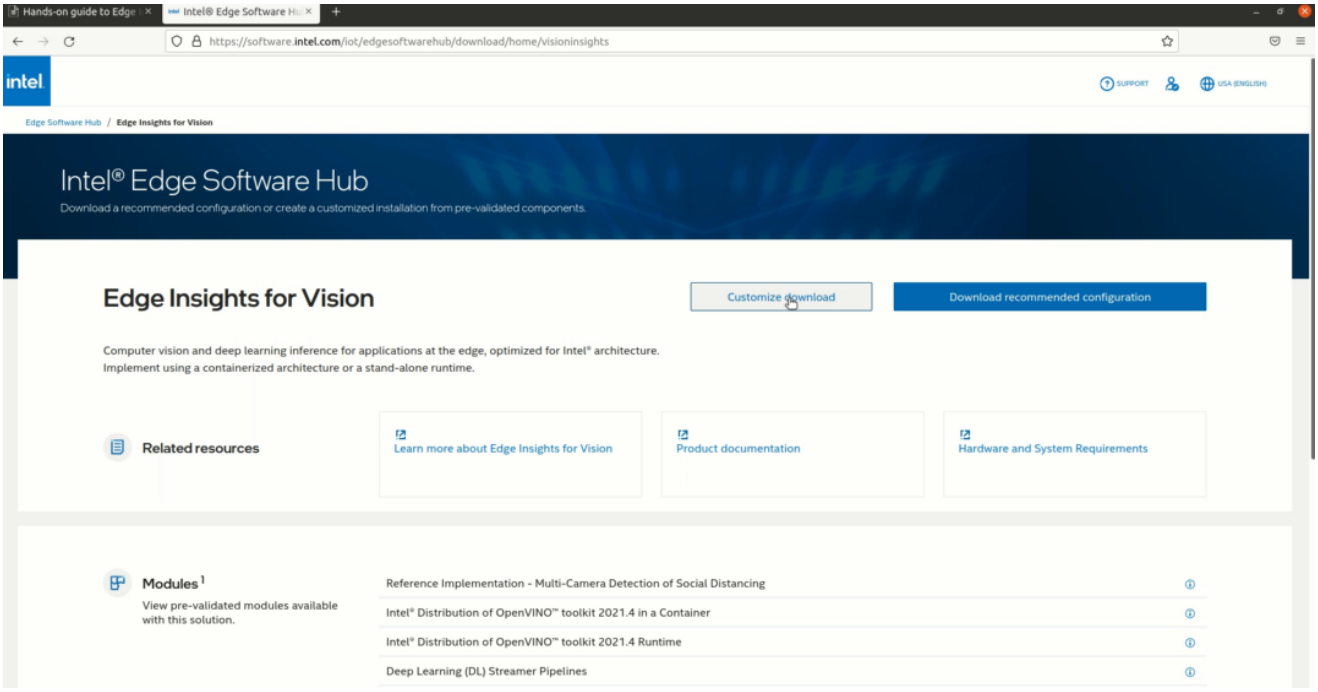
Step 2 : Configure Edge Insight for Vision package on ESH

- Go to Intel ESH website: https://software.intel.com/content/www/us/en/develop/topics/iot/edge-solutions/vision-recipes.html (https://software.intel.com/content/www/us/en/develop/topics/iot/edge-solutions/vision-recipes.html)

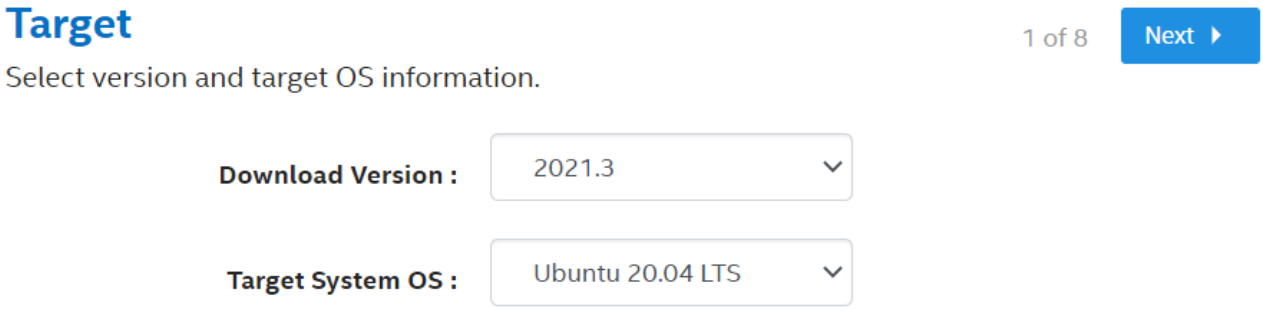
- Configure and download required components:



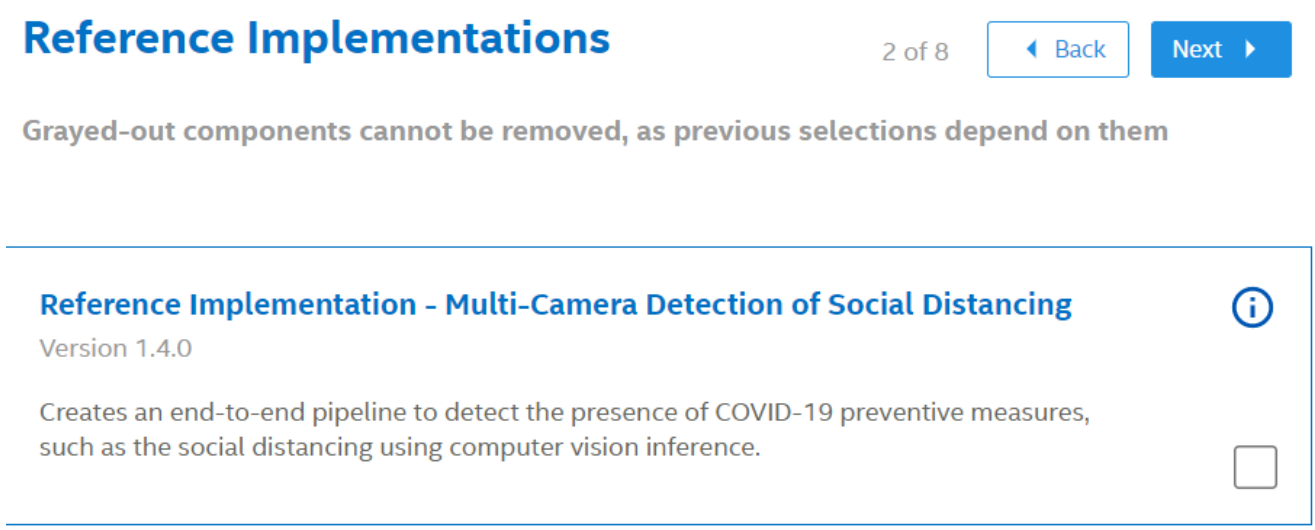
- Select “Customize download”



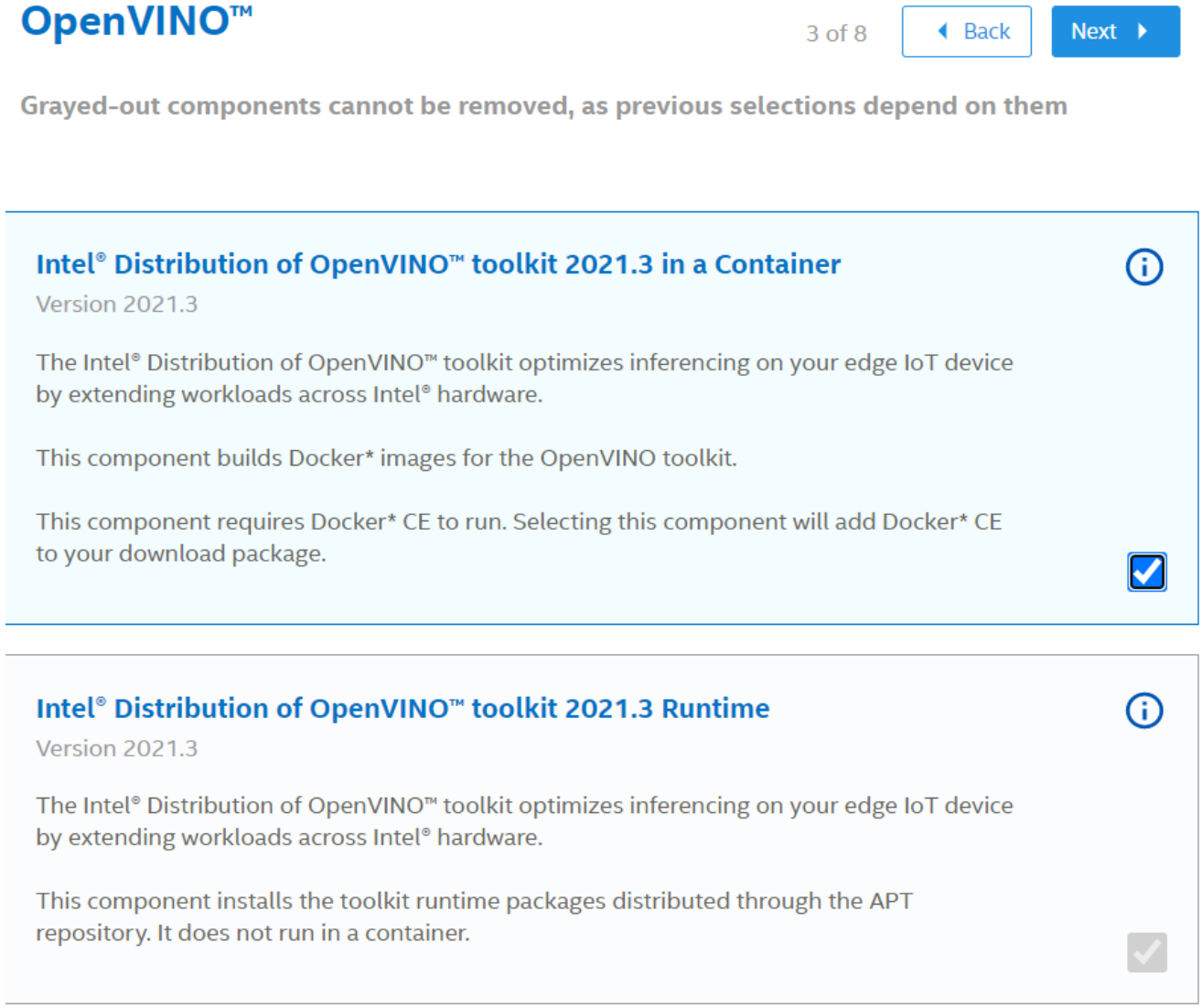
- Select Openvino Download version: 2021.3 for Target System OS



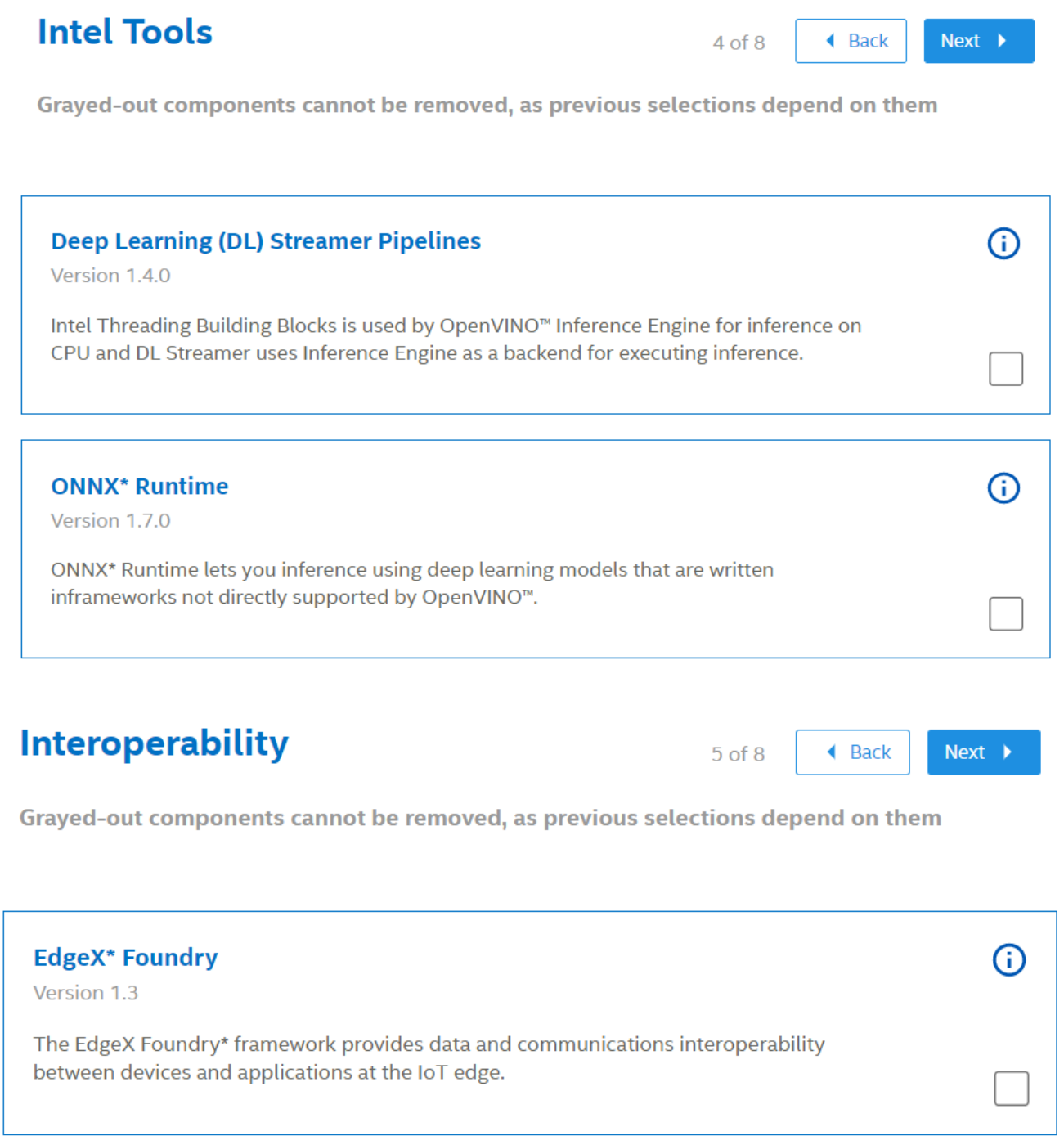
- Select the Multi-Camera Detection if need to detect social distancing



- Select “Intel OpenVINO 2021.3 in a Container”



- Select other tools if needed



Container Engines & Orchestration

6 of 8 | Back | Next

Grayed-out components cannot be removed, as previous selections depend on them

Docker Compose*

Version 1.24.0

Docker* Compose is a tool for defining and running multi-container Docker* applications. It lets you run and manage multiple containers at the same time with a single command. This module requires Docker* CE to run. Selecting this module will add Docker* CE to your download package.

☐

Docker Community Edition (CE)*

Version 20.10.5

Docker* is a container framework. It lets you package an application and its dependencies in a virtual container, making it easier to create, deploy, run, and manage applications.

☑

K3s* (Lightweight Kubernetes*)

Version 1.20.4

K3s is a lightweight Kubernetes distribution providing automated container management

Cloud Edge Connectors

7 of 8 | Back | Next

Grayed-out components cannot be removed, as previous selections depend on them

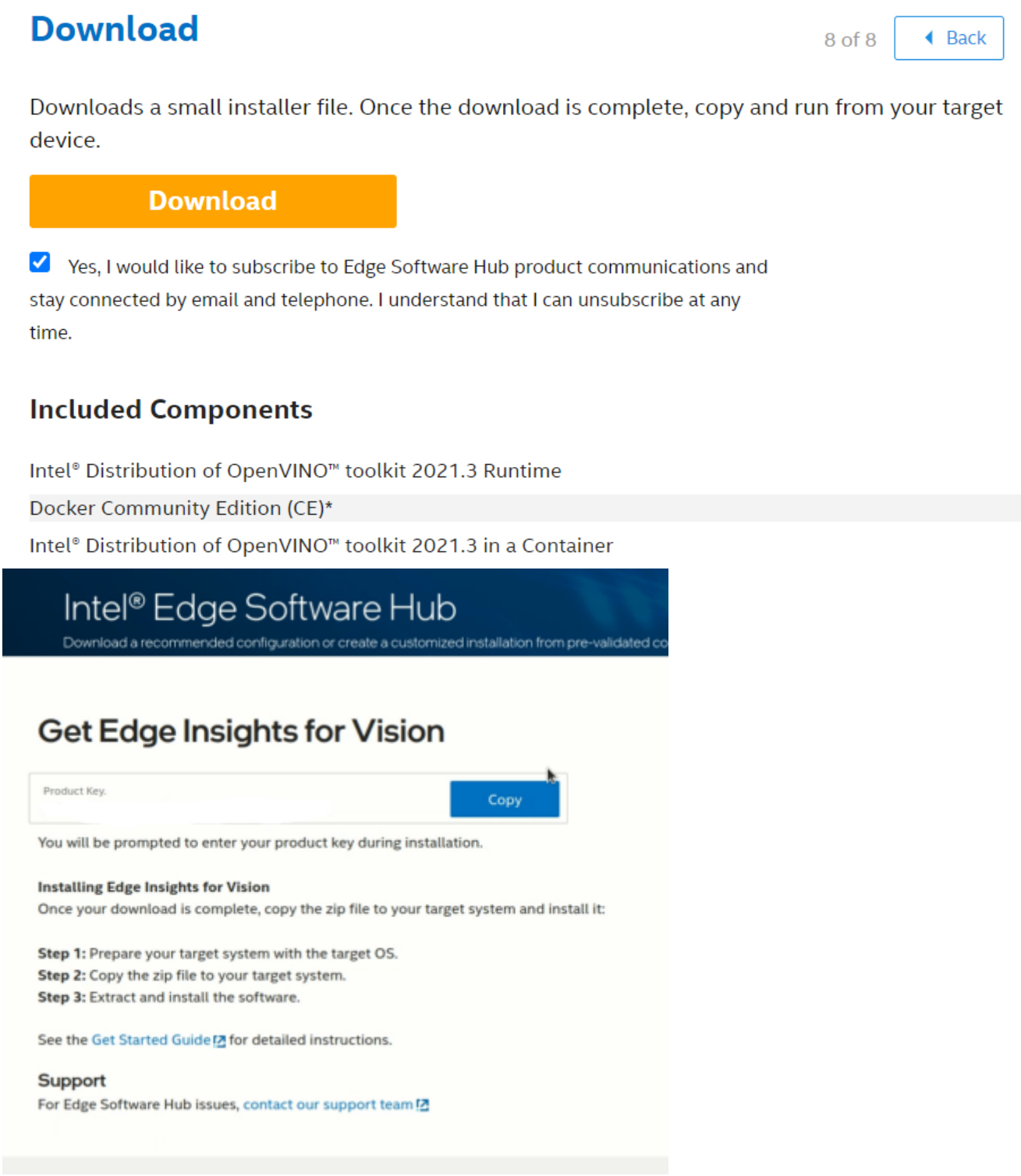
Amazon Web Services Greengrass Prerequisites

Version 1.11

AWS* IoT Greengrass Prerequisites installs AWS IoT Greeengrass Core, AWS IoT Device SDK and AWS IoT Greengrass SDK. These enable local data collection, analysis, and communication for your AWS-connected edge devices.

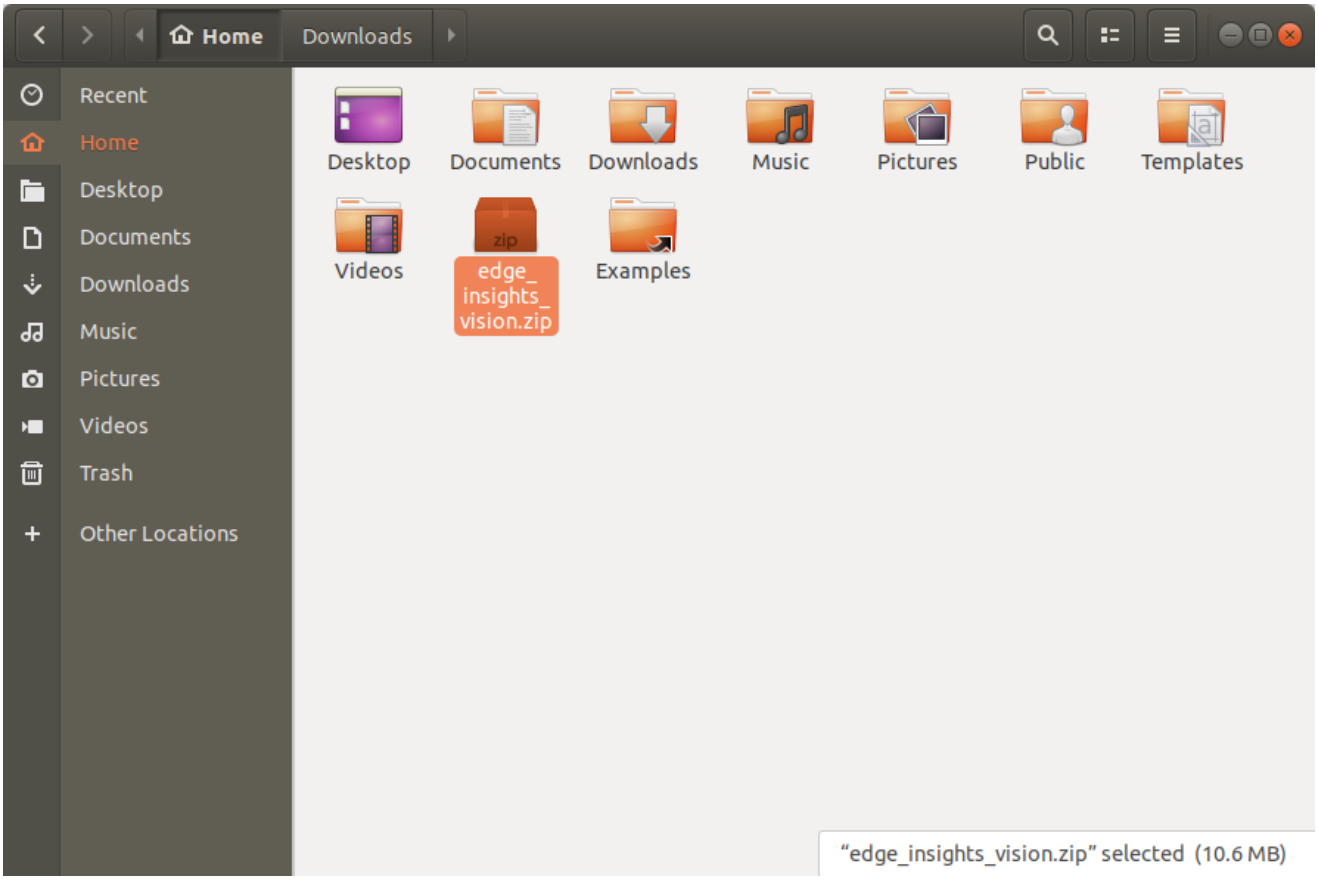
☐

- Download the package and copy the .zip file to target device and copy the product key

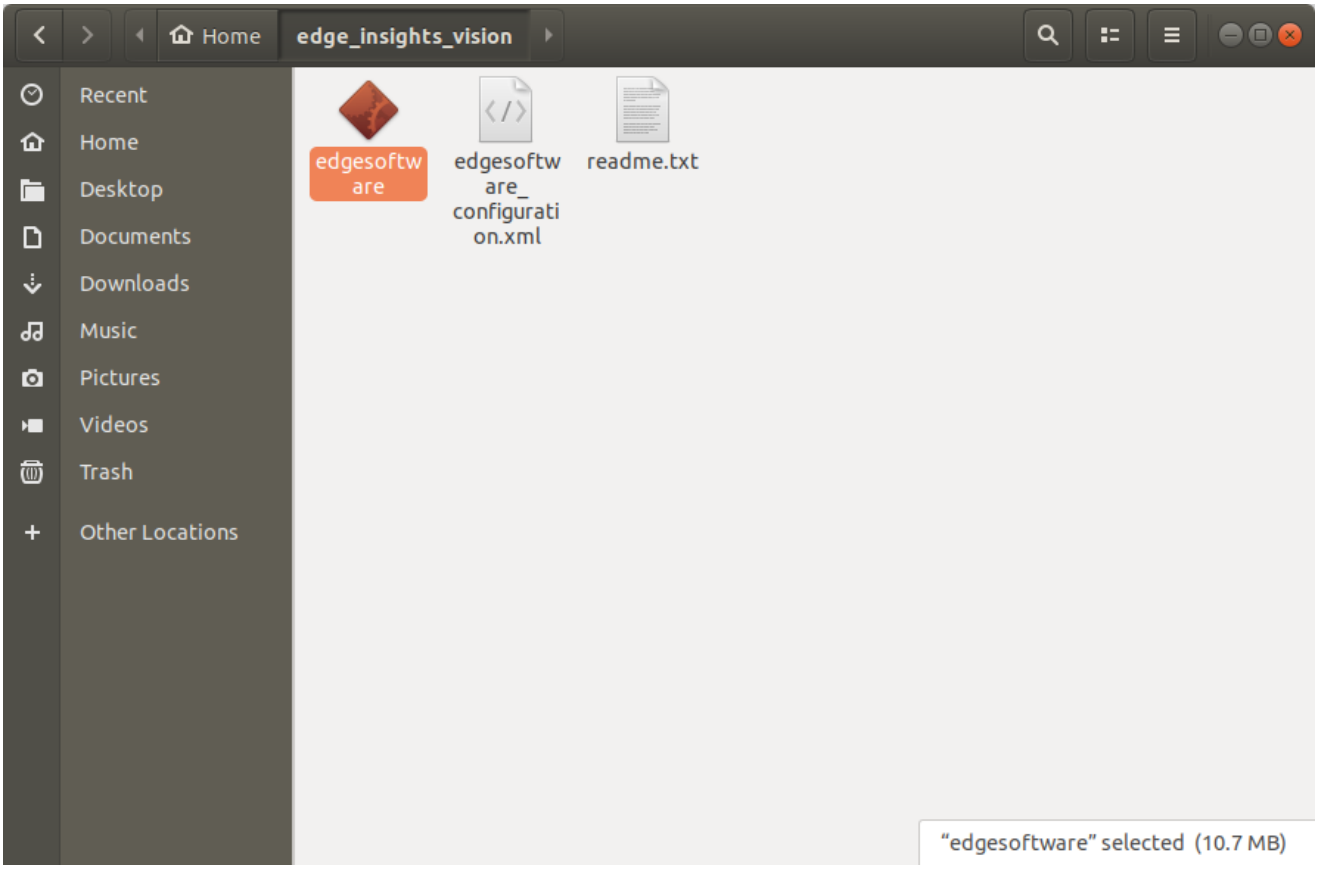


Step 3 : Install on target device

- Copy the Edge Insights for Vision .zip File to the Target System



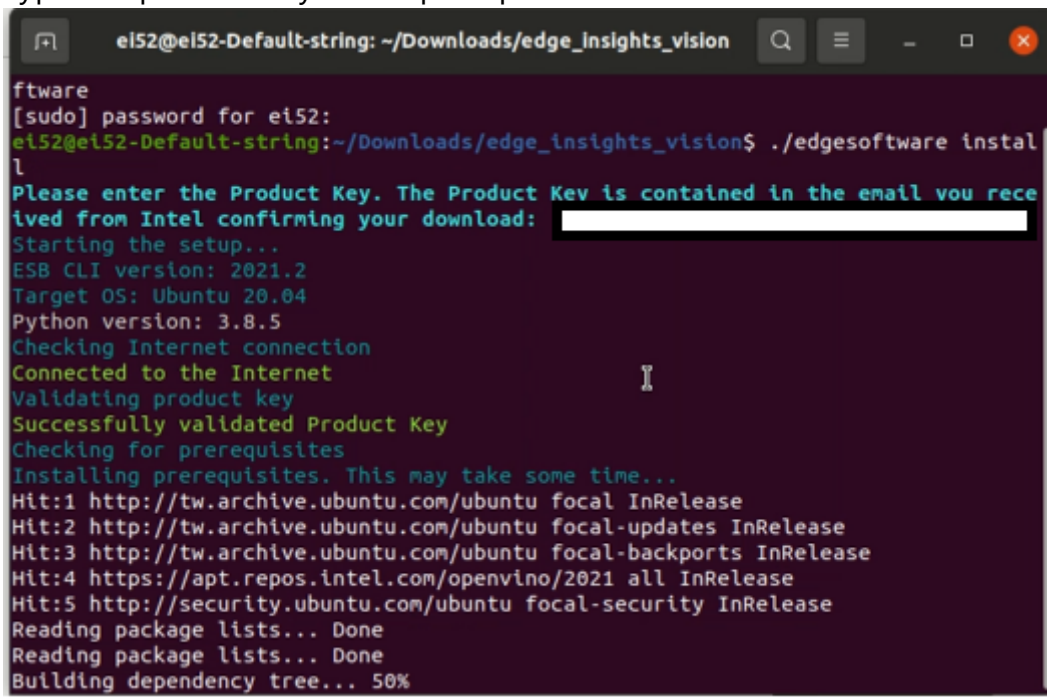
- Extract the Edge Insights for Vision Software



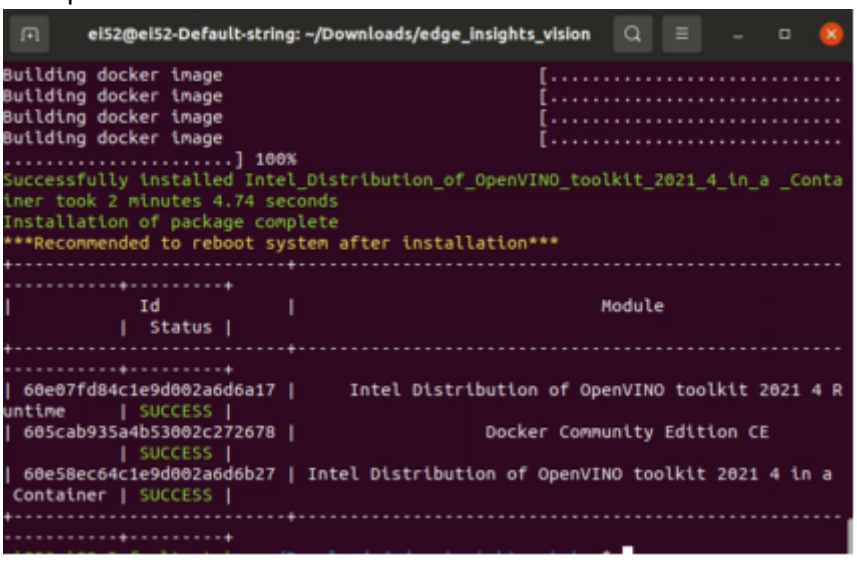
- Install the Edge Insights for Vision Software
- Run these commands

```
$ cd edge_insights_vision/
$ sudo chmod 775 edgesoftware
$ ./edgesoftware install
```

- Type the product key at the prompt:



- When the installation is complete, you will see the message Installation of package complete and the installation status for each module.



Step 4 : Pull docker image from docker hub & run smart city demo

- To get the video streaming file for demo

```
$ wget -O ~/Downloads/NewVideo2.mp4 \
    https://github.com/incluit/OpenVino-For-SmartCity\
    /raw/master/data/NewVideo2.mp4
```

- Pull image from Docker Hub

```
$ docker pull docker_hub_destination/openvino:2021.3_developer_models
```

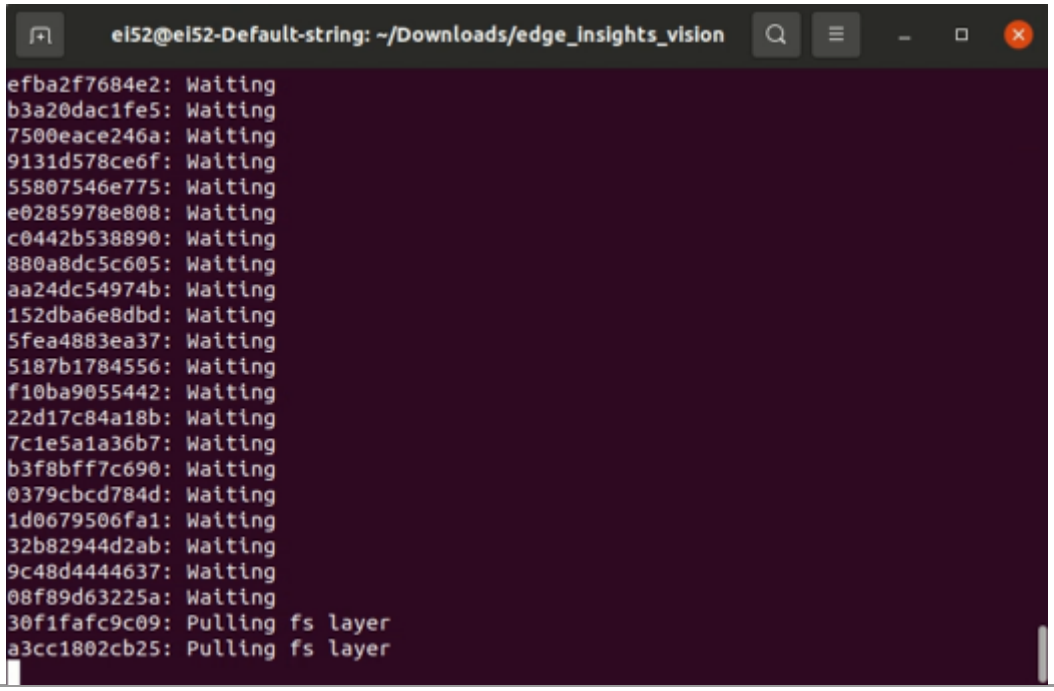
NOTE: “docker_hub_destination” should be changed to existing resource locations: sertek or synnexgrp or wpig or wt1com

- Run smart city demo

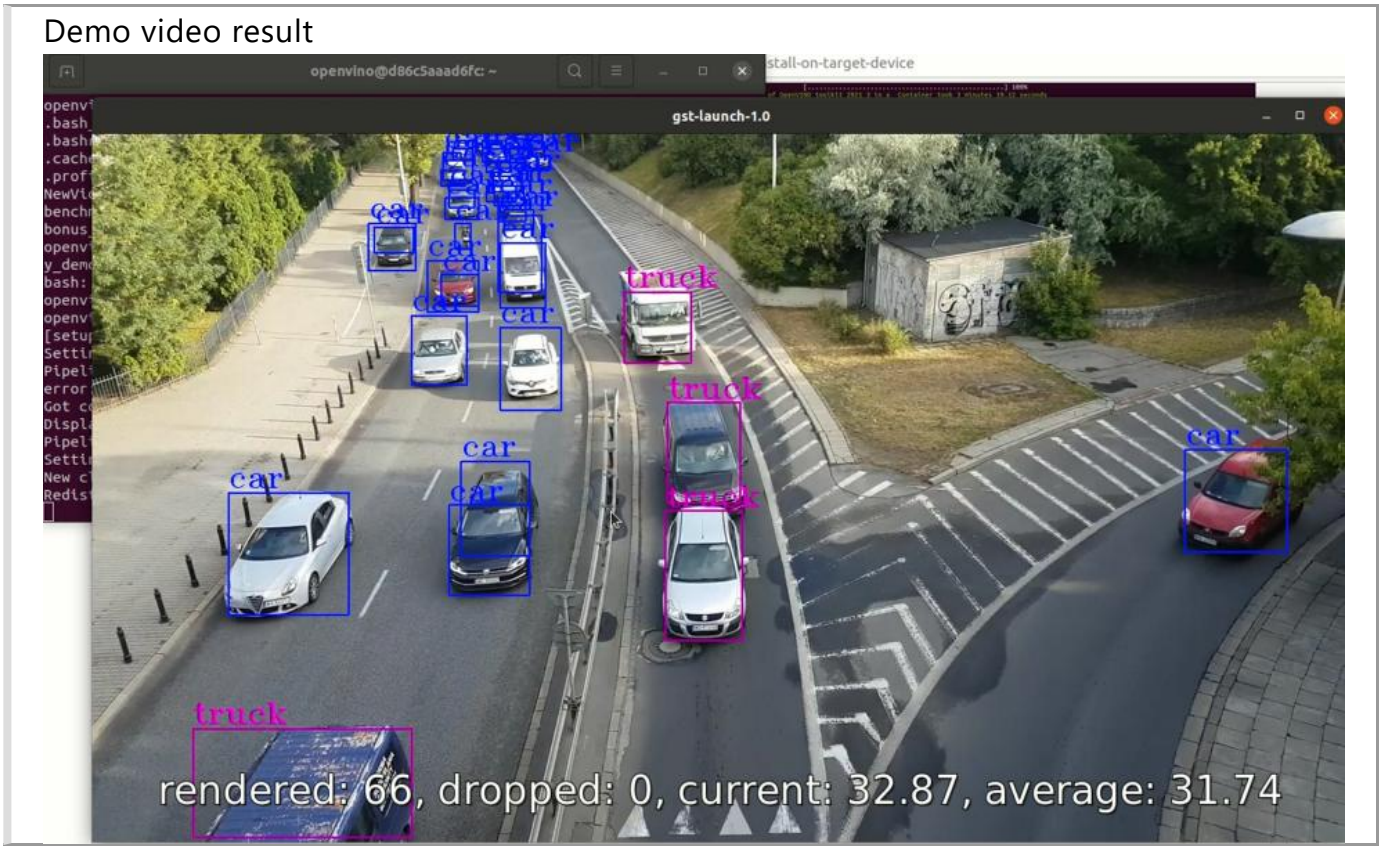
```
$ docker run -it -v /tmp/.X11-unix:/tmp/.X11-unix \
    -e DISPLAY=$DISPLAY -v ~/Downloads:/mnt \
    --device /dev/dri:/dev/dri \
    --group-add=$(stat -c "%g" /dev/dri/render*) \
    --rm docker_hub_destination/openvino:2021.3_developer_models
$ cd /home/openvino/
$ ./smartcity_demo.sh
```

NOTE: “docker_hub_destination” should be changed to existing resource locations: sertek or synnexgrp or wpig or wt1com

pull docker image



Demo video result



Step 5 : Run benchmark application with AI models (yolov3 & yolov4)

- Run Benchmark_app to measure the performance on device

```
$ docker run -it -v /tmp/.X11-unix:/tmp/.X11-unix \
    -e DISPLAY=$DISPLAY -v ~/Downloads:/mnt \
    --device /dev/dri:/dev/dri \
    --group-add=$(stat -c "%g" /dev/dri/render*) \
    --rm docker_hub_destination/openvino:2021.3_developer_models
$ cd /home/openvino/
$ python3 run_command.py
```

NOTE: “docker_hub_destination” should be changed to existing resource locations: sertek or synnexgrp or wpig or wt1com

- Benchmark results:

- Intel® Celeron® 6305E

AI Model	Device	Throughput(FPS)		
		FP32	FP16	INT8
yolo-v3-tf	CPU	1.54	1.54	5.59
	GPU	7.43	14.62	30.03
yolo-v4-tf	CPU	0.72	0.72	2.12
	GPU	3.36	6.84	13.83

- Intel® Core™ i3-1115G4E

AI Model	Device	Throughput(FPS)		
		FP32	FP16	INT8
yolo-v3-tf	CPU	3.49	3.43	12.66
	GPU	7.53	14.75	29.42
yolo-v4-tf	CPU	1.51	1.51	4.68
	GPU	3.34	6.85	13.79

- Intel® Core™ i5-1145G7E

AI Model	Device	Throughput(FPS)		
		FP32	FP16	INT8
yolo-v3-tf	CPU	5.38	5.43	20.47
	GPU	11.39	22.24	47.59
yolo-v4-tf	CPU	2.57	2.56	8.19
	GPU	4.71	9.77	21.18

<Model name: YOLOv3(FP32/FP16/INT8); Target device: CPU>

```
cmd:python3 /opt/intel/openvino/deployment_tools/tools/benchmark_tool/benchmark_app.py -m /opt/intel/
openvino_models/public/yolo-v3-tf/FP32/yolo-v3-tf.xml -d CPU -api async -t 60

        MKLDNNPlugin............ version 2.1
[ INFO ] Read network took 627.51 ms
[ INFO ] Load network took 966.85 ms
[Step 10/11] Measuring performance (Start inference asynchronously, 4 inference requests using 4 stre
ams for CPU, limits: 60000 ms duration)
[ INFO ] First inference took 290.47 ms
Latency:    729.22 ms
Throughput: 5.38 FPS
cmd:python3 /opt/intel/openvino/deployment_tools/tools/benchmark_tool/benchmark_app.py -m /opt/intel/
openvino_models/public/yolo-v3-tf/FP16/yolo-v3-tf.xml -d CPU -api async -t 60

        MKLDNNPlugin............ version 2.1
[ INFO ] Read network took 321.03 ms
[ INFO ] Load network took 1006.28 ms
[Step 10/11] Measuring performance (Start inference asynchronously, 4 inference requests using 4 stre
ams for CPU, limits: 60000 ms duration)
[ INFO ] First inference took 299.34 ms
Latency:    734.35 ms
Throughput: 5.43 FPS
cmd:python3 /opt/intel/openvino/deployment_tools/tools/benchmark_tool/benchmark_app.py -m /opt/intel/
openvino_models/public/yolo-v3-tf/FP16-INT8/yolo-v3-tf.xml -d CPU -api async -t 60

        MKLDNNPlugin............ version 2.1
[ INFO ] Read network took 178.49 ms
[ INFO ] Load network took 969.19 ms
[Step 10/11] Measuring performance (Start inference asynchronously, 4 inference requests using 4 stre
ams for CPU, limits: 60000 ms duration)
[ INFO ] First inference took 81.29 ms
Latency:    196.14 ms
Throughput: 20.47 FPS
```

<Model name: YOLOv3(FP32/FP16/INT8); Target device: GPU>

```
cmd:python3 /opt/intel/openvino/deployment_tools/tools/benchmark_tool/benchmark_app.py -m /opt/intel/
openvino_models/public/yolo-v3-tf/FP32/yolo-v3-tf.xml -d GPU -api async -t 60

        clDNNPlugin............ version 2.1
[ INFO ] Read network took 99.09 ms
[ INFO ] Load network took 39962.91 ms
[Step 10/11] Measuring performance (Start inference asynchronously, 4 inference requests using 2 stre
ams for GPU, limits: 60000 ms duration)
[ INFO ] First inference took 87.04 ms
Latency:    346.26 ms
Throughput: 11.39 FPS
cmd:python3 /opt/intel/openvino/deployment_tools/tools/benchmark_tool/benchmark_app.py -m /opt/intel/
openvino_models/public/yolo-v3-tf/FP16/yolo-v3-tf.xml -d GPU -api async -t 60

        clDNNPlugin............ version 2.1
[ INFO ] Read network took 54.54 ms
[ INFO ] Load network took 39796.82 ms
[Step 10/11] Measuring performance (Start inference asynchronously, 4 inference requests using 2 stre
ams for GPU, limits: 60000 ms duration)
[ INFO ] First inference took 47.62 ms
Latency:    179.70 ms
Throughput: 22.24 FPS
cmd:python3 /opt/intel/openvino/deployment_tools/tools/benchmark_tool/benchmark_app.py -m /opt/intel/
openvino_models/public/yolo-v3-tf/FP16-INT8/yolo-v3-tf.xml -d GPU -api async -t 60

        clDNNPlugin............ version 2.1
[ INFO ] Read network took 41.15 ms
[ INFO ] Load network took 45651.05 ms
[Step 10/11] Measuring performance (Start inference asynchronously, 4 inference requests using 2 stre
ams for GPU, limits: 60000 ms duration)
[ INFO ] First inference took 24.94 ms
Latency:    84.14 ms
Throughput: 47.59 FPS
```

<Model name: YOLOv4(FP32/FP16/INT8); Target device: CPU>

```
cmd:python3 /opt/intel/openvino/deployment_tools/tools/benchmark_tool/benchmark_app.py -m /opt/intel/
openvino_models/public/yolo-v4-tf/FP32/yolo-v4-tf.xml -d CPU -api async -t 60

        MKLDNNPlugin............ version 2.1
[ INFO ] Read network took 645.99 ms
[ INFO ] Load network took 1016.38 ms
[Step 10/11] Measuring performance (Start inference asynchronously, 4 inference requests using 4 stre
ams for CPU, limits: 60000 ms duration)
[ INFO ] First inference took 607.19 ms
Latency:    1550.43 ms
Throughput: 2.57 FPS
cmd:python3 /opt/intel/openvino/deployment_tools/tools/benchmark_tool/benchmark_app.py -m /opt/intel/
openvino_models/public/yolo-v4-tf/FP16/yolo-v4-tf.xml -d CPU -api async -t 60

        MKLDNNPlugin............ version 2.1
[ INFO ] Read network took 331.68 ms
[ INFO ] Load network took 1101.35 ms
[Step 10/11] Measuring performance (Start inference asynchronously, 4 inference requests using 4 stre
ams for CPU, limits: 60000 ms duration)
[ INFO ] First inference took 604.96 ms
Latency:    1542.77 ms
Throughput: 2.56 FPS
cmd:python3 /opt/intel/openvino/deployment_tools/tools/benchmark_tool/benchmark_app.py -m /opt/intel/
openvino_models/public/yolo-v4-tf/FP16-INT8/yolo-v4-tf.xml -d CPU -api async -t 60

        MKLDNNPlugin............ version 2.1
[ INFO ] Read network took 191.63 ms
[ INFO ] Load network took 1082.36 ms
[Step 10/11] Measuring performance (Start inference asynchronously, 4 inference requests using 4 stre
ams for CPU, limits: 60000 ms duration)
[ INFO ] First inference took 200.93 ms
Latency:    481.49 ms
Throughput: 8.19 FPS
```

<Model name: YOLOv4(FP32/FP16/INT8); Target device: GPU>

```
cmd:python3 /opt/intel/openvino/deployment_tools/tools/benchmark_tool/benchmark_app.py -m /opt/intel/
openvino_models/public/yolo-v4-tf/FP32/yolo-v4-tf.xml -d GPU -api async -t 60

        clDNNPlugin............ version 2.1
[ INFO ] Read network took 98.28 ms
[ INFO ] Load network took 57381.43 ms
[Step 10/11] Measuring performance (Start inference asynchronously, 4 inference requests using 2 stre
ams for GPU, limits: 60000 ms duration)
[ INFO ] First inference took 216.27 ms
Latency:    843.79 ms
Throughput: 4.71 FPS
cmd:python3 /opt/intel/openvino/deployment_tools/tools/benchmark_tool/benchmark_app.py -m /opt/intel/
openvino_models/public/yolo-v4-tf/FP16/yolo-v4-tf.xml -d GPU -api async -t 60

        clDNNPlugin............ version 2.1
[ INFO ] Read network took 55.40 ms
[ INFO ] Load network took 56599.55 ms
[Step 10/11] Measuring performance (Start inference asynchronously, 4 inference requests using 2 stre
ams for GPU, limits: 60000 ms duration)
[ INFO ] First inference took 106.79 ms
Latency:    409.35 ms
Throughput: 9.77 FPS
cmd:python3 /opt/intel/openvino/deployment_tools/tools/benchmark_tool/benchmark_app.py -m /opt/intel/
openvino_models/public/yolo-v4-tf/FP16-INT8/yolo-v4-tf.xml -d GPU -api async -t 60

        clDNNPlugin............ version 2.1
[ INFO ] Read network took 46.52 ms
[ INFO ] Load network took 65023.63 ms
[Step 10/11] Measuring performance (Start inference asynchronously, 4 inference requests using 2 stre
ams for GPU, limits: 60000 ms duration)
[ INFO ] First inference took 52.70 ms
Latency:    189.05 ms
Throughput: 21.18 FPS
```

Step 6: Run benchmark application with other AI models

If you wish to run the benchmark with other AI models, run the following command and change the “route_to_model” to the destination of the model file and “model.xml” to the model file name.

```
$  python3 \
    /opt/intel/openvino/deployment_tools/tools/benchmark_tool/benchmark_app.py \
    -m /mnt/path_to_model/model.xml \
    -d CPU -api async -t 60
```

NOTE: There is some pre-build model in the “/opt/intel/openvino_models/public” you can try to use these models.

Below is an example using resnet-50-tf public model

```
$ python3 \
   /opt/intel/openvino/deployment_tools/tools/benchmark_tool/benchmark_app.py \
   -m /opt/intel/openvino_models/public/resnet-50-tf/FP16-INT8/resnet-50-tf.xml \
   -d CPU -api async -t 60
```

(Optional) SOP for using VPU in Docker Container

Step 1 : Download HDDL driver package

You can use the official Intel® Distribution of OpenVINO™ toolkit packages from trusted resources.
See more on the product page (https://software.intel.com/content/www/us/en/develop/tools/openvino-toolkit/choose-download.html).

Or you can download a small archive of HDDL driver package from Amazon Web Services*. Available releases for Ubuntu* 18.04:

- 2020.2 (https://storage.openvinotoolkit.org/drivers/vpu/hddl/2020.2/hddl_ubuntu18_1076.tgz)
- 2020.3 (https://storage.openvinotoolkit.org/drivers/vpu/hddl/2020.3/hddl_ubuntu18_1167.tgz)
- 2020.3.1 (https://storage.openvinotoolkit.org/drivers/vpu/hddl/2020.3.1/hddl_ubuntu18_1409.tgz)
- 2020.3.2 (https://storage.openvinotoolkit.org/drivers/vpu/hddl/2020.3.2/hddl_ubuntu18_1651.tgz)
- 2020.4 (https://storage.openvinotoolkit.org/drivers/vpu/hddl/2020.4/hddl_ubuntu18_1229.tgz)
- 2021.1 (https://storage.openvinotoolkit.org/drivers/vpu/hddl/2021.1/hddl_ubuntu18_1380.tgz)
- 2021.2 (https://storage.openvinotoolkit.org/drivers/vpu/hddl/2021.2/hddl_ubuntu18_1509.tgz)
- 2021.3 (https://storage.openvinotoolkit.org/drivers/vpu/hddl/2021.3/hddl_ubuntu18_1636.tgz)
- 2021.4 (https://storage.openvinotoolkit.org/drivers/vpu/hddl/2021.4/hddl_ubuntu18_1701.tgz)

Step 2 : Install HDDL Driver

- Run following commands on the host machine with HDDL device from <archive_extract_folder>/hddl:

```
$ sudo -i
$ source setupvars.sh
$ ./install_IVAD_VPU_dependencies.sh
$ reboot
$ sudo -i
$ source setupvars.sh && ./bin/hddldaemon -d
```

NOTE: Please don’t close the terminal which runing the hddldaemon.

Step 3 : Using VPU in Docker Container

- Run Benchmark_app with VPU

```
$ docker run -it -v /tmp/.X11-unix:/tmp/.X11-unix \
    -e DISPLAY=$DISPLAY -v ~/Downloads:/mnt \
    --device /dev/dri:/dev/dri \
    --group-add=$(stat -c "%g" /dev/dri/render*) \
    --rm docker_hub_destination/openvino:2021.3_developer_models
$ cd /home/openvino/
$ python3 \
    /opt/intel/openvino/deployment_tools/tools/benchmark_tool/benchmark_app.py \
    -m /opt/intel/openvino_models/public/resnet-50-tf/FP16/resnet-50-tf.xml \
    -d HDDL -api async -t 60
```

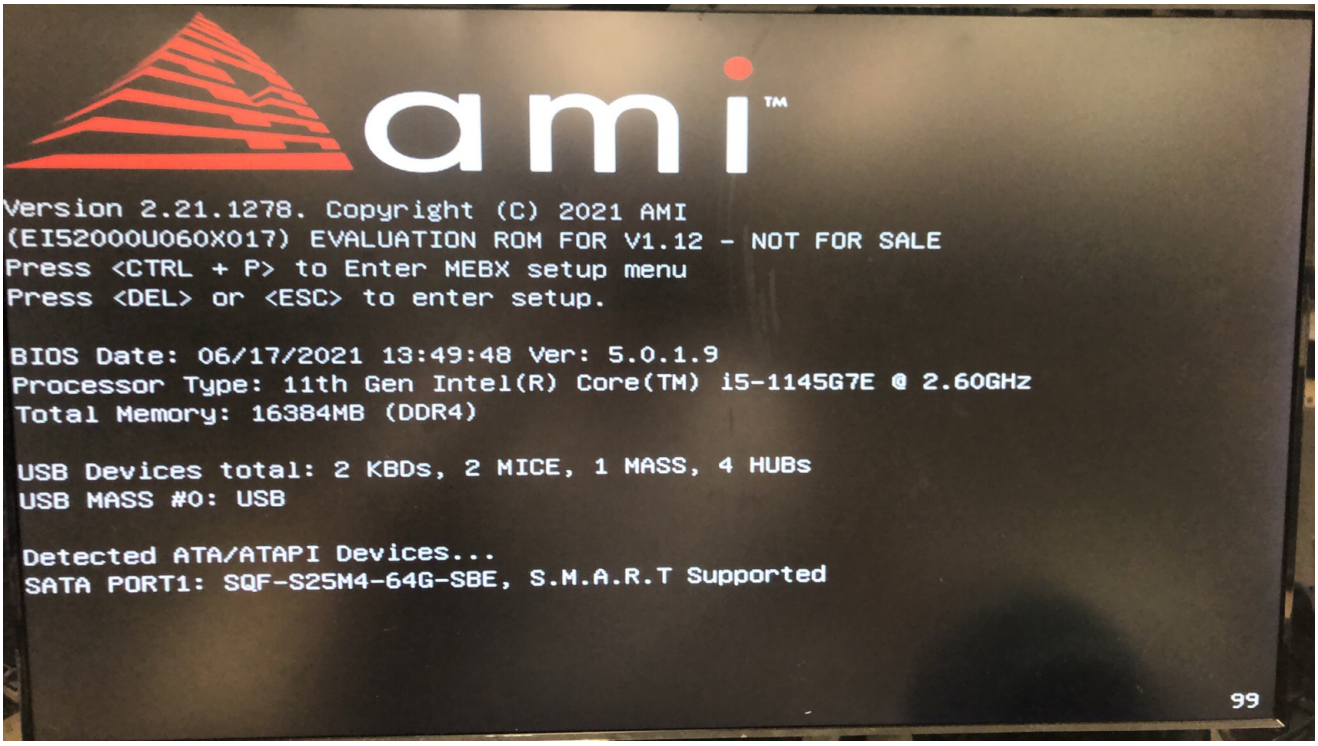
NOTE: “docker_hub_destination” should be changed to existing resource locations: sertek or synnexgrp or wpig or wt1com

(Optional) SOP for turn on Turbo Mode

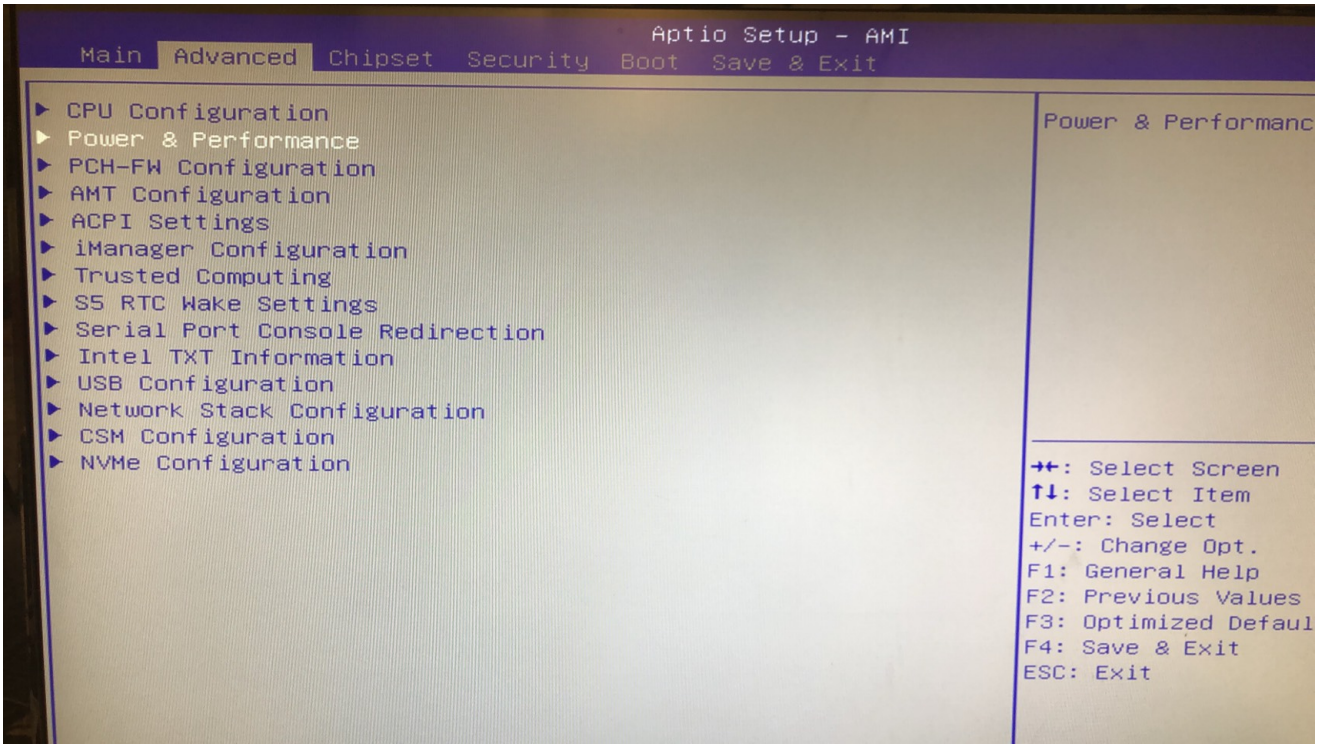
> refer to the following guide for performance optimization.

Step 1 : Configuration to get best CPU performance

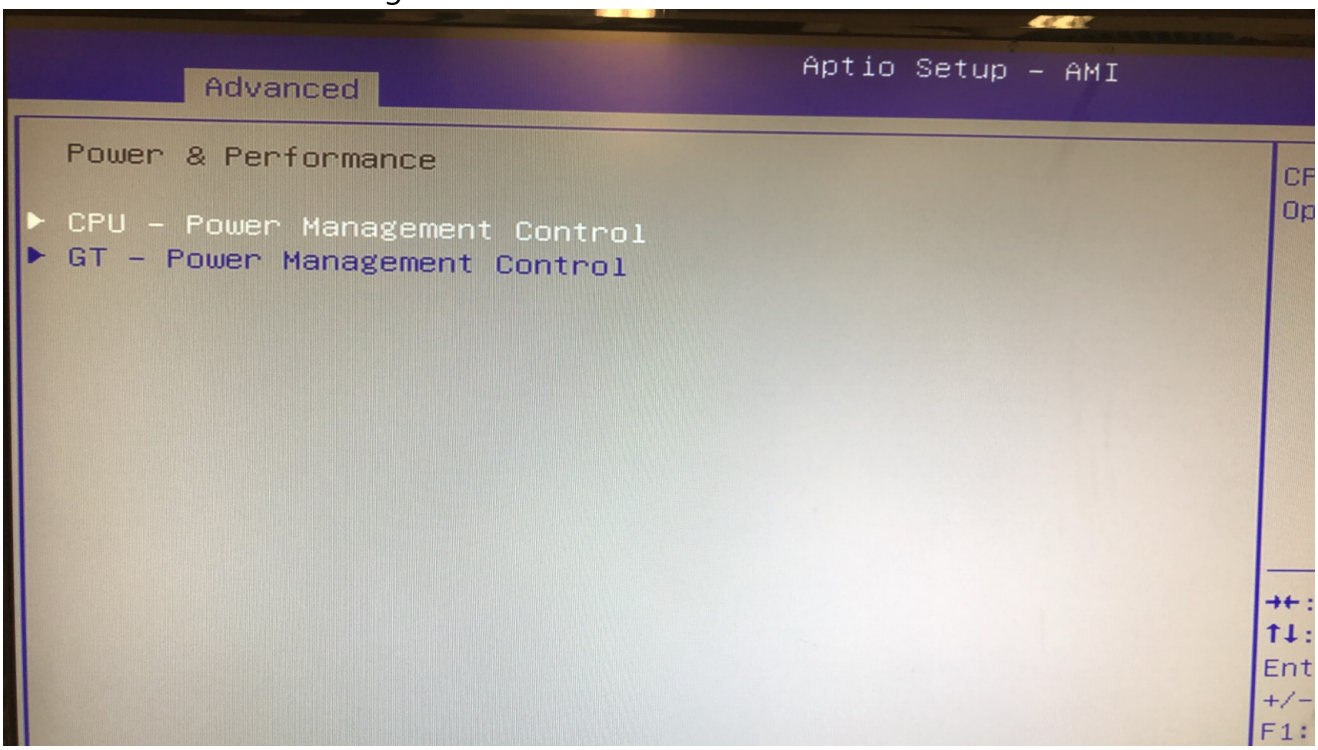
- Press “Delete” to go to BIOS Setting



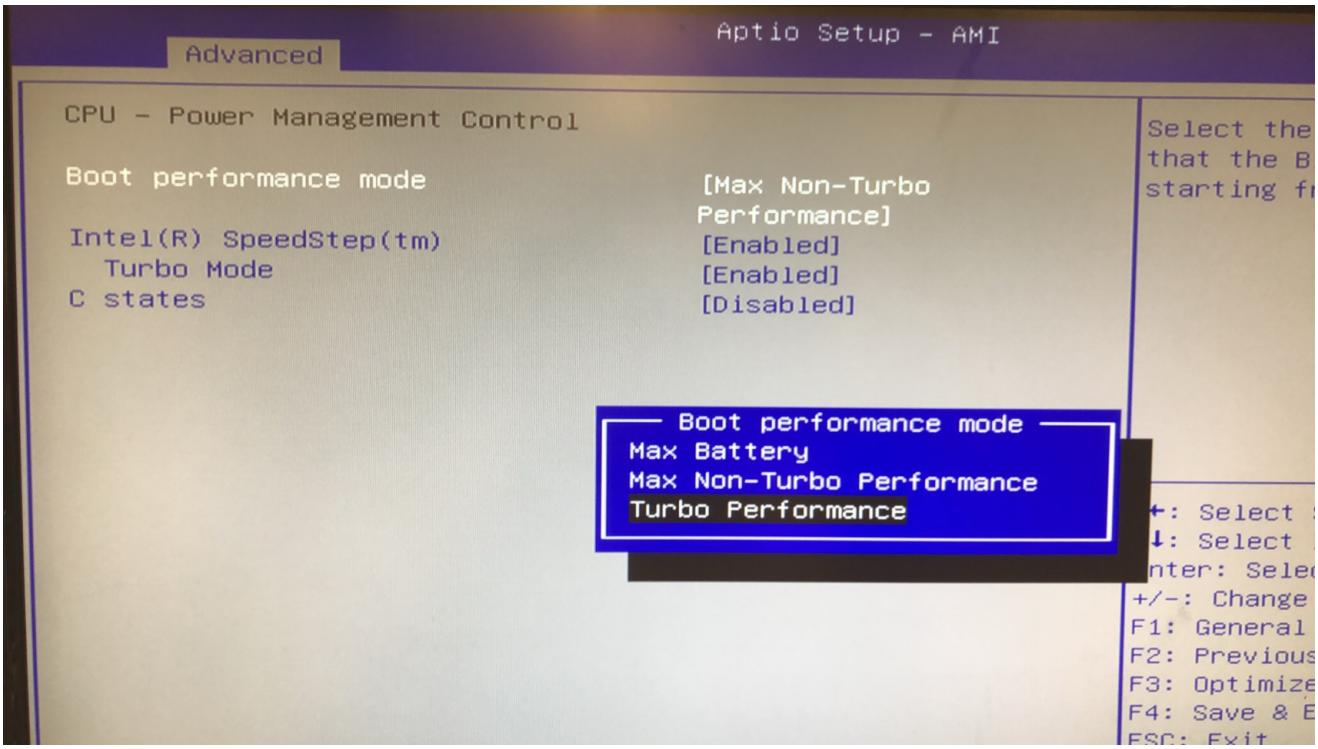
- Go to “Advanced” tab and select the “Power & Performance”



- Go to “CPU - Power Management Control”

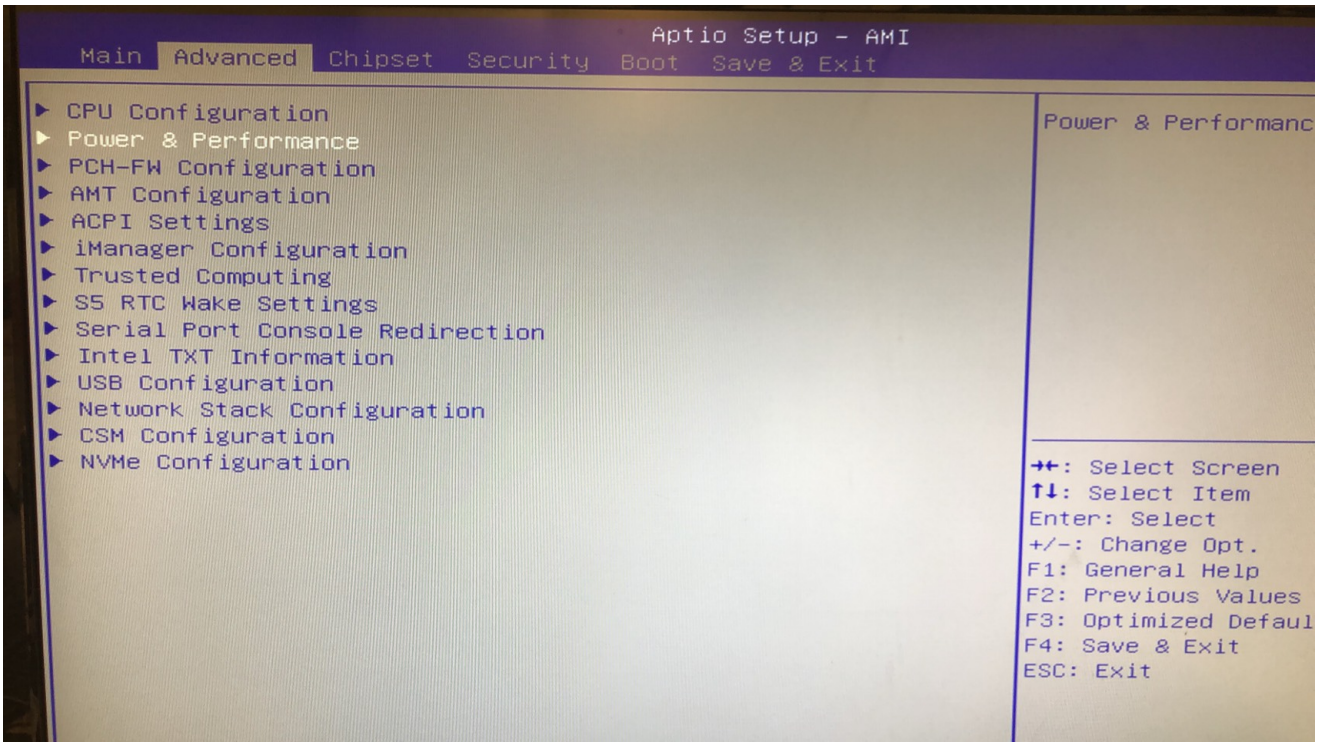


- Select the “Boot performance mode” modify it to “Turbo Performance”(Default:Max Non-Turbo Performance)

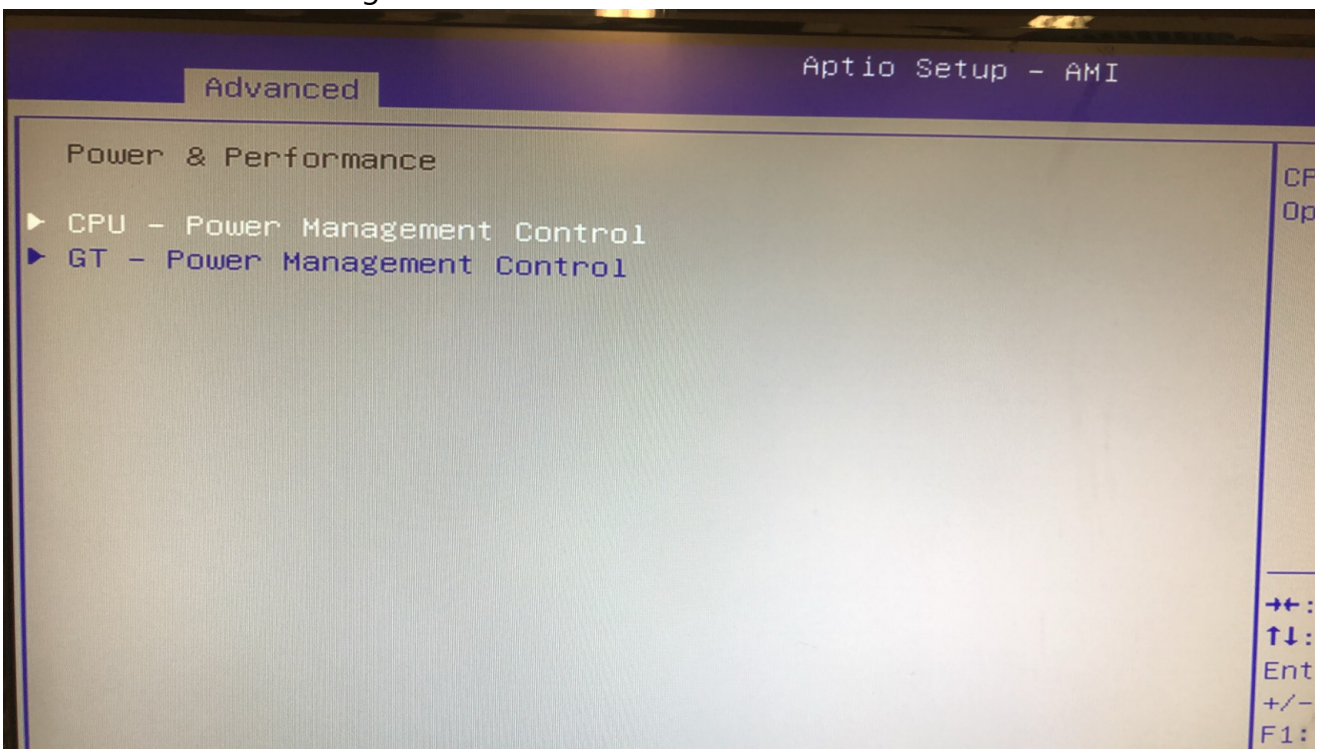


Step 2 : Configuration to get best GPU performance

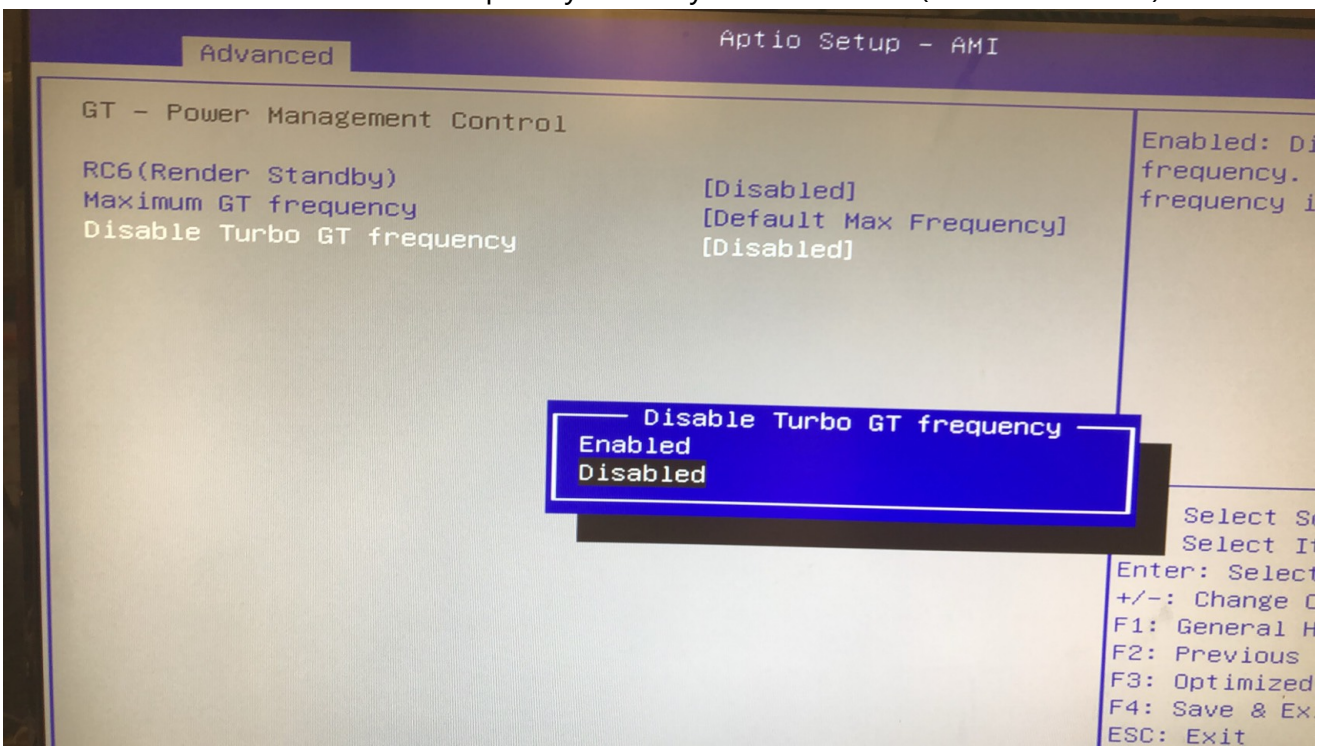
- Go to “Advanced” tab and select the “Power & Performance”



- Go to “GT - Power Management Control”



- Select the “Disable Turbo GT frequency” modify it to “Disable”(Default:Disable)



- Press “F4” to Save the configuration.