



**VMSS 2.0**

**New Generation of Aupera Video  
Machine Learning Streaming  
Server**

**Quick Start User Guide**

**Document Revision: 3.0.0**

1	RUNNING AUPERA VMSS2.0 ON VMACCEL .....	4
	1.1 LAUNCHING AUPERA VMSS2.0 INSTANCE ON VCK5000 .....	4
2	RUNNING AUPERA VMSS2.0 PIPELINES THROUGH WEB CLIENT ....	6
	2.1 ACCESSING VMSS2.0 WEB CLIENT .....	6
	2.2 ADD VIDEO STREAMS TO ACCESS APPLICATIONS .....	6
	2.3 RUNNING AUPERA'S PROPRIETARY PIPELINES .....	8
	2.3.1 RUNNING AUPERA'S CROWD FLOW APPLICATION .....	8
	2.4 BUILDING YOUR OWN (CUSTOM) PIPELINES .....	17
	2.5 MODEL ZOO EVALUATION .....	19
	2.6 RUNNING YOUR OWN (CUSTOM) PIPELINE .....	22
	2.7 USING VIDEO STREAMS TO ADD NEW STREAM, PLAY VIDEOS, OR CHANGE SNAPSHOTS .....	27
	2.8 AI HOSTS PAGE .....	30
	2.9 USEFUL TOOLS .....	33
	2.10 LAUNCHING YOUR OWN RTSP STREAMS .....	36
3	RUNNING AUPERA VMSS2.0 PIPELINES ON SERVER (VIA COMMAND LINE) .....	37
	3.1 ACCESSING VMSS2.0 SERVER DOCKER .....	37
	3.2 RUNNING VMSS2.0 PIPELINES .....	39
	3.3 PIPELINE EXAMPLES .....	40



# 1 RUNNING AUPERA VMSS2.0 ON VMACCEL

In this section, we will introduce the steps to sign-up for the VMAccel demo account, access Aupera Video Machine-learning Streaming Server 2.0 (VMSS2.0) instance and launch custom RTSP streams for tasks. Users can go through these steps to open the door to using the functionalities of Aupera VMSS2.0 AI pipelines in later sections.

## 1.1 Launching Aupera VMSS2.0 instance on VCK5000

Please sign up for a demo account at: <https://www.vmaccel.com/vmssdemo>

After completing the sign-up form, you will receive an email with your demo credentials. Please follow the instructions in the email to log into VMAccel. Please note that this account will allow you to evaluate VMSS2.0 for free for 5 hours.

**NOTE:** The trial accounts allow 5 hours of free evaluation of VMSS2.0. The trial accounts are currently configured to automatically delete any instances when a user logs out. You may use a total of 5 hours of runtime, and this could be extended over several days. Your account will be locked once you have depleted 5 hours of runtime.

For each user, a VMAccel instance will be automatically launched. Once you login, under *Project->Compute* you should see your instance by selecting “Instances” in the left-hand side bar. You should be able to see the instance that was just created for you being spawned as shown in Figure 1.1.

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
Aupera VMSS2.0 Demo	Aupera Technologies VMSS2.0	184.105.10.122	VCK5000-PQ.1 (8G-64-128)	default	Active	us-east-1a	None	Running	35 minutes	Create Snapshot

**Figure 1.1. VMAccel instances main page**

After some time, the status of the instance will change to Active. At this point, the installation of the VMSS2.0 server and the client on the instance begin. Please allow about 3 minutes for this process to finish after you see the status has changed to Active.

VMSS 2.0 consists of two major modules, Aupera Video AI Client (AVAC) and Aupera Video AI Server (AVAS). AVAC allows users to use a user-friendly GUI to connect to AVAS. Additionally, more advanced users may work with AVAS via the command line directly. More detailed information would be introduced in later chapters.

**NOTE 1:** We may use Aupera Video AI Client (AVAC) and Aupera's VMSS2.0 AI Client interchangeably throughout this document. We may also use Aupera Video AI Server (AVAS) and Aupera's VMSS2.0 AI Server interchangeably throughout this document.

**NOTE 2:** To run VMSS2.0 pipelines, you can either use Aupera's VMSS2.0 Web Client (as described in **Section 2**) or access the VMSS2.0 server (as described in **Section 3**) using the command line.

**NOTE 3:** To facilitate testing, the VMAccel instance will automatically start three RTSP streams:

```
rtsp://<vmaccel_instance_ip_address>:8554/retail  
rtsp://<vmaccel_instance_ip_address>:8554/car  
rtsp://<vmaccel_instance_ip_address>:8554/crowd  
rtsp://<vmaccel_instance_ip_address>:8554/imagenet
```

**vmaccel\_instance\_ip\_address** is the IP address that is shown in the image above under IP Address when you are in *Project->Compute->Instances*. The first stream contains a crowded scene of people passing by and is the most useful for testing head, person, and other human related detections. We use this video for benchmarking our crowd applications. The second stream contains objects usually encountered in a retail scenario. We use this stream for throughput benchmarking.

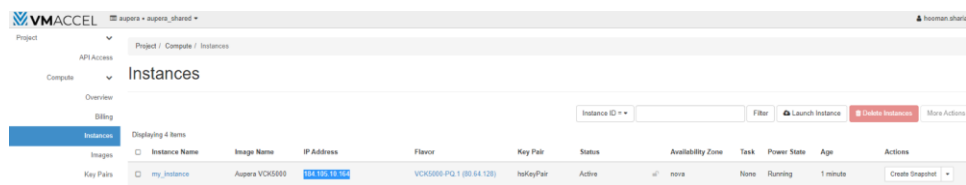
Additionally, AVAC, the VMSS2.0 Web Client, can connect to any RTSP streams that are broadcast on open ports.

## 2 RUNNING AUPERA VMSS2.0 PIPELINES THROUGH WEB CLIENT

In this section, we will describe the steps to run the Aupera VMSS2.0 pipelines through the VMSS2.0 Web Client. It is mainly divided into three subsections: accessing the Aupera VMSS2.0 Web Client, running the crowd flow application, and launching custom pipelines. Users can use RTSP streams to run different tasks through the Web Client interface and easily view the task running results.

### 2.1 Accessing VMSS2.0 Web Client

After launching an Aupera VMSS2.0 instance on VMAccel, you can access the web client by copying the IP address of your instance into your browser on your local machine; and adding the port 3004. For example, in the screenshot below, the IP address of the instance is 184.105.10.164 which means that the web client can be accessed by typing <http://184.105.10.164:3004/> in your browser.



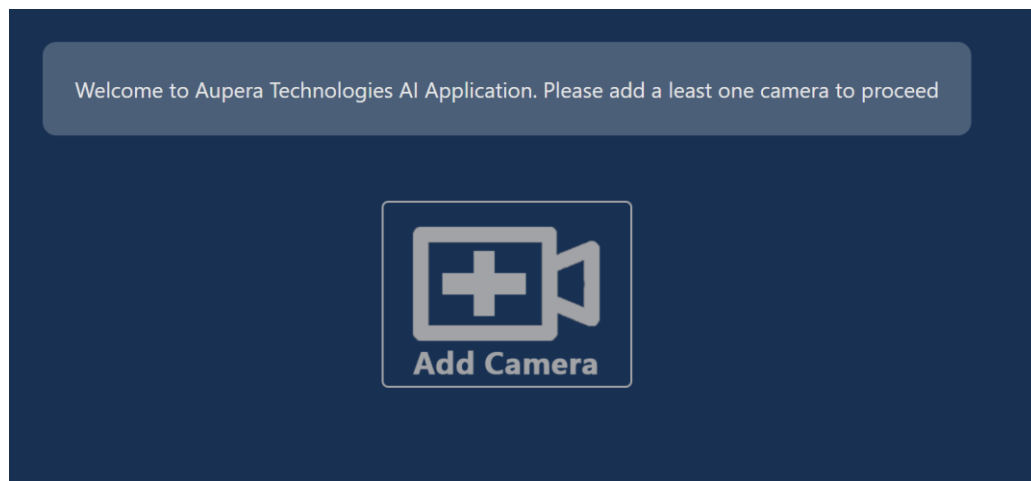
Instance ID	Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
my_instance	Aupera VMSS2.0	my_image	184.105.10.164	VMSS2.0-FG 1 (8G 64 GB)	my_keypair	Active	us-east-1	None	Running	1 minute	Create Snapshot

Figure 2.2. VMAccel instances page with instance IP address highlighted

**NOTE:** Since the relevant ports of your VMAccel instance are open to the public, you do not need to use the browser of the VMAccel instance (accessible through VNC) to launch the Web Client. You can use the browser on any machine with an internet connection.

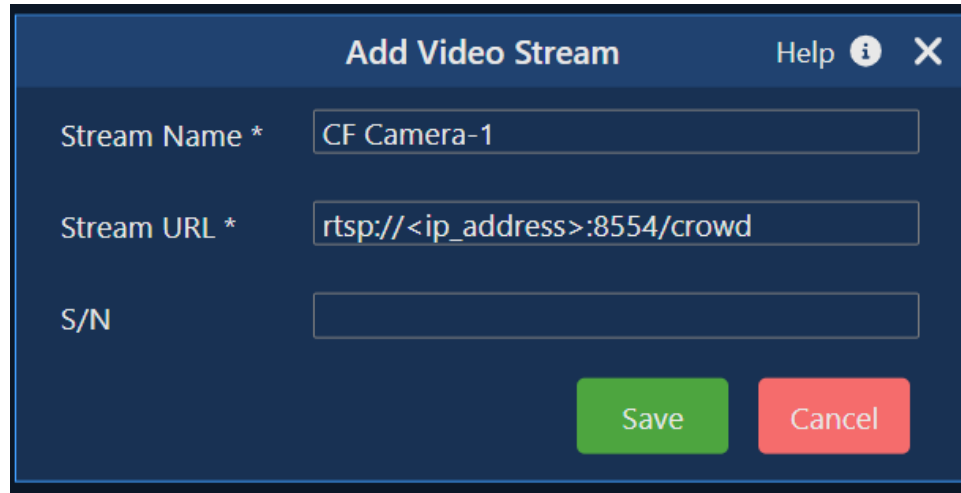
### 2.2 Add Video Streams to Access Applications

A. Click **Add Stream** to add at least one video stream to access the Application functions.



**Figure 2.3. Aupera web application page**

B. Enter a **Name** (any arbitrary name can be used) and **URL** (**S/N** is not required). **Make sure that RTSP URL is correct.** You can click the vertical ellipsis (three dots) icon on the added video stream to access the video stream configuration shown in Figure 2.4.



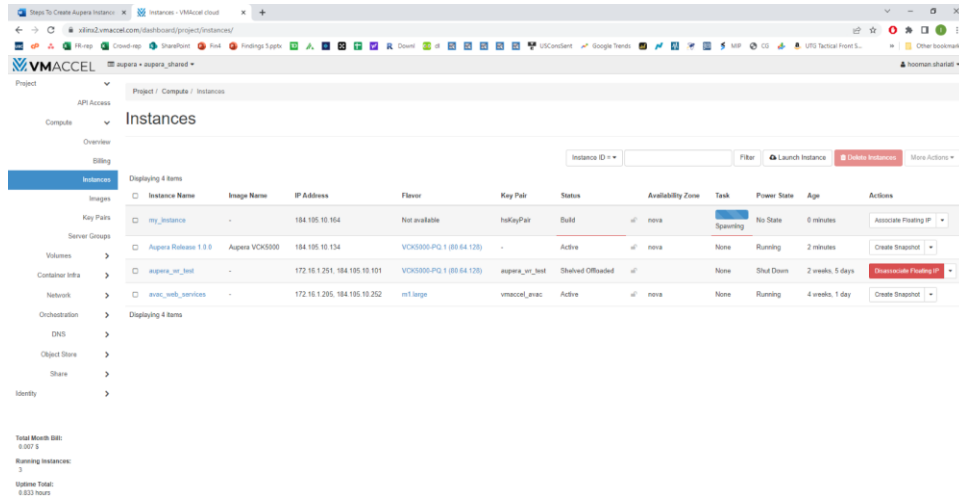
**Figure 2.4. Aupera web application page – add video stream**

**NOTE 1:** You can use either your own publicly available RTSP stream (perhaps the one you launched using the instructions in Section 2.2); or it can be one of the three RTSP streams that automatically started up when you launched your VMAccel instance.

For the latter you can use one of the following addresses:

**rtsp://<vmaccel\_instance\_ip\_address>:8554/crowd**  
**rtsp://<vmaccel\_instance\_ip\_address>:8554/retail**  
**rtsp://<vmaccel\_instance\_ip\_address>:8554/car**  
**rtsp://<vmaccel\_instance\_ip\_address>:8554/imagenet**

Just make sure to replace the **vmaccel\_instance\_ip\_address** with the IP address of the VMAccel instance you just launched. You can find this IP address by clicking on the left-hand sidebar select “Instances” you can see the instance you just created being spawned as shown in Figure 2.5.



**Figure 2.5. VMACcel instances page – preparing a new instance**

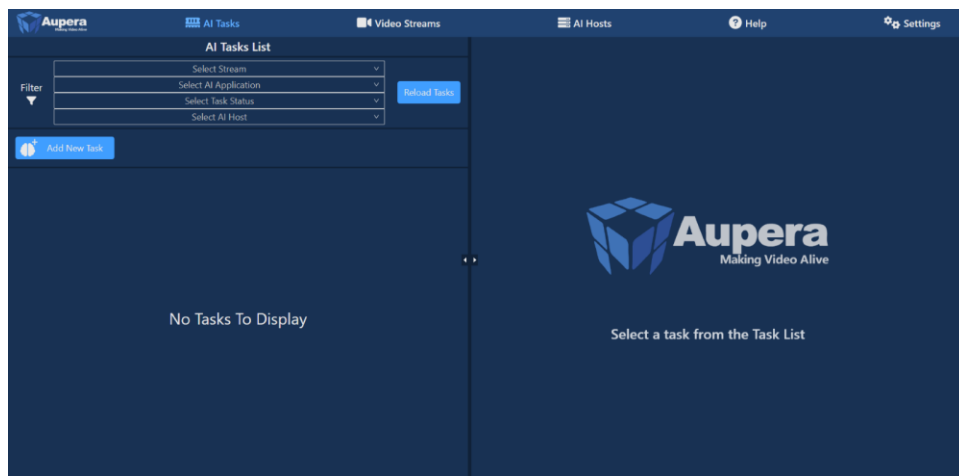
Again, the first stream shows a scene with people (used for crowd, person, and person attributed applications) while the second stream shows a scene with retail objects.

**NOTE 2:** If you need help in this step, click **Help** button in Figure 2.4 for more details about each input field and the next steps.

## 2.3 Running Aupera's Proprietary Pipelines

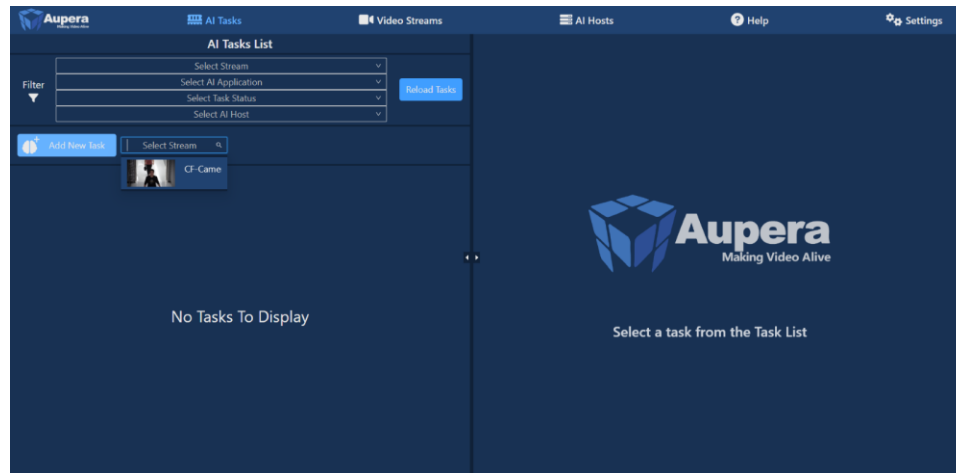
### 2.3.1 Running Aupera's Crowd Flow Application

A. Click **AI Tasks Hub**, then click **Add New Task** button, then there will be a drop down to select a video stream



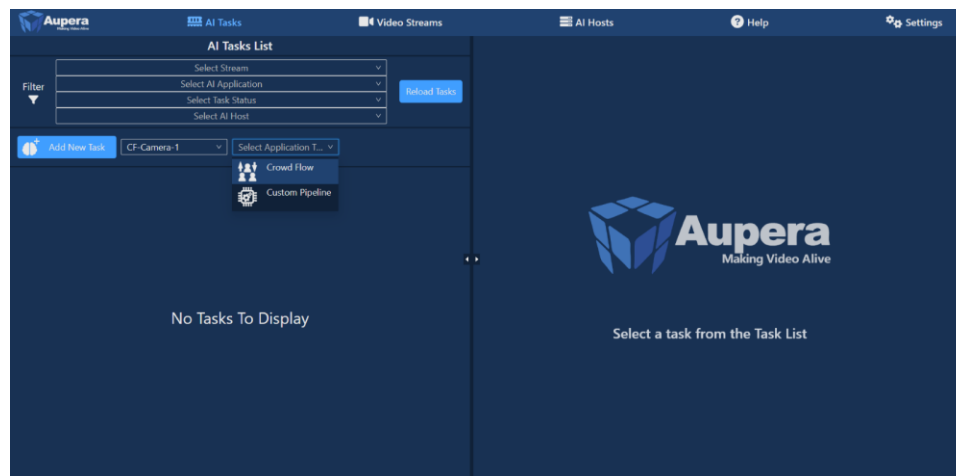
**Figure 2.6. Aupera web application page – AI Tasks Hub Page**





**Figure 2.7 Aupera web application page – AI Tasks Hub Page, Add New Task button clicked**

B. After selecting a Video Stream, another drop down will appear to select an application. Select Crowd Flow to open Crowd Flow application set up controls.



**Figure 2.8 Aupera web application page – AI Tasks Hub Page, Camera Selected**

C. After selecting Crowd Flow, another dropdown will appear to select an AI Host. Only AI hosts supporting the application selected in Application dropdown (see point above) will be displayed in this dropdown. Most of the time the Web Application will have a default AI host from the beginning. More AI Hosts may be added on the AI Hosts page. (For more information on AI Hosts page, please go to **2.2.5 AI Hosts**)

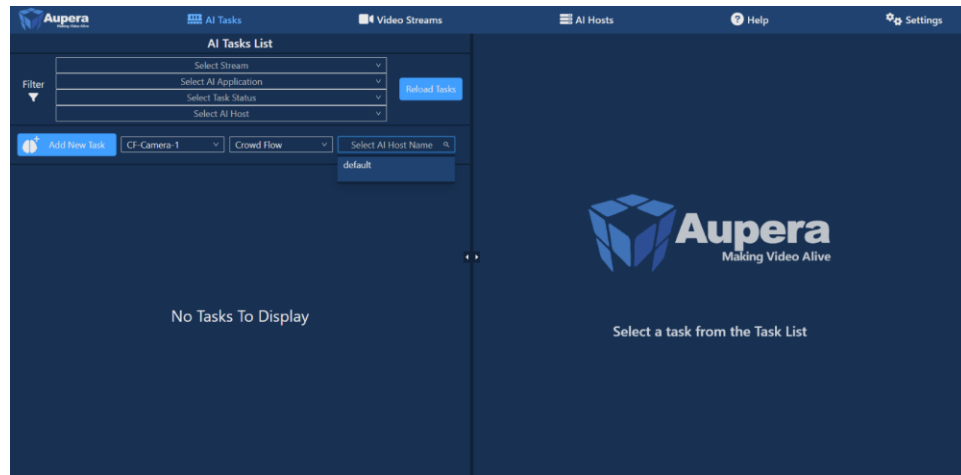


Figure 2.9 Aupera web application page – AI Tasks Hub Page, AI Host Selected

D. After Stream, Application, and AI Hosts are selected, the Crowd Flow Task Setup window will appear. (This is the right panel of the AI Tasks page extended, collapsing this window by clicking **Task List** or **cancel** button will return to the previous page)

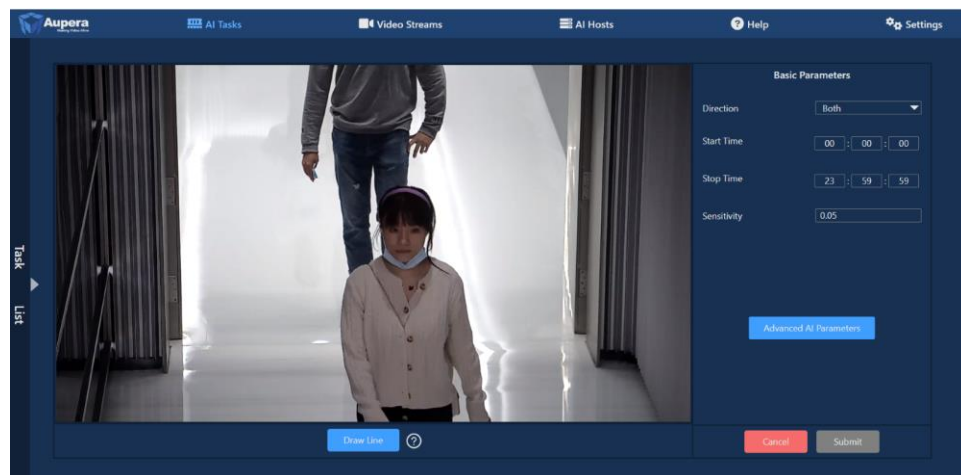


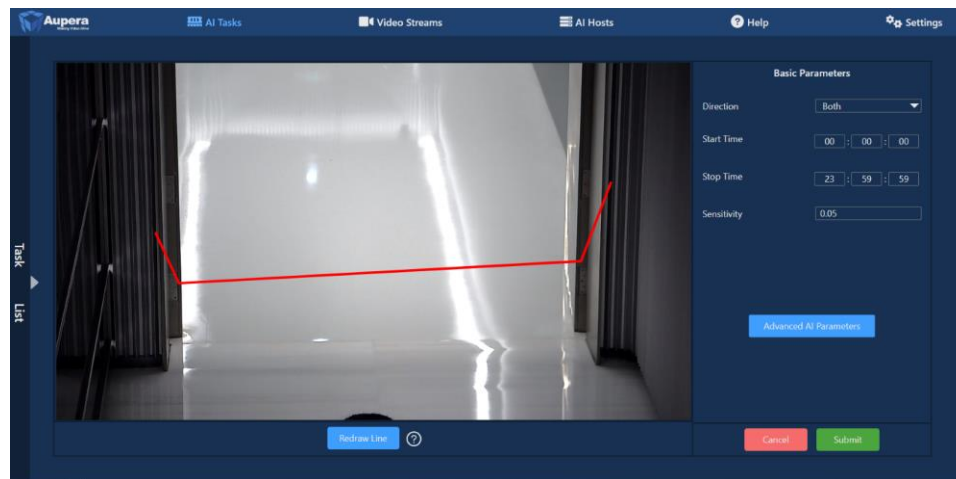
Figure 2.10. Aupera web application page – crowd flow task creation

E. You need to draw a line to indicate a border for people crossing. When a person's head crosses the line, IN/OUT count will reflect this event. These border lines may consist of up to a maximum of 14 segments, a less segmented line provides more accurate results. In and Out directions are determined by the way a line was drawn. If line is drawn from left to right, Movement from top to bottom will be registered as "IN", bottom to top - "OUT".

1. To start drawing, click the **Draw Line** button, the cursor will change to a cross.
2. **Left-click** and **hold the mouse button** on the place where you would like to start the line.
3. Drag the Line to the place where you want to finish the first segment, then release the mouse button.
4. Move a mouse to the end of the next segment to complete it.
5. To finish drawing, click the **Right mouse button**, unfinished segment will be deleted.

\* To see an animation of how this is done, click the help button beside **Draw** button

After the Line is drawn, “Draw Line” will change to “Redraw Line”. Click button if you want to delete the Line drawn and start drawing from the beginning. It is recommended to draw U shaped lines as shown in Figure 2.11 to capture people who may move parallel to the line and around it.

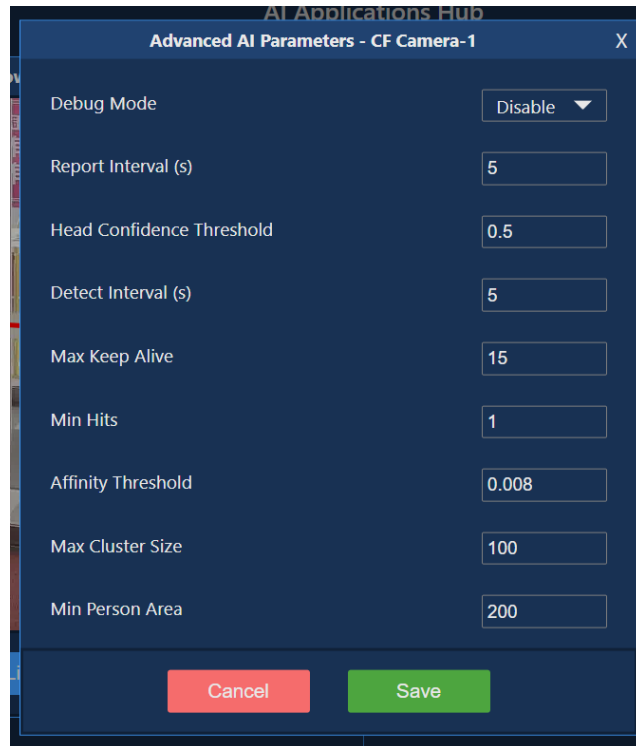


**Figure 2.11. Aupera web application page – crowd flow task lines drawing example**

F. Changing Basic Parameters is not required to start the task, you can keep default values.

1. **Direction** – Count people going “In”, “Out” (“Entering” / “Exiting”) or both directions
2. **Start/Stop Time** – counting will be started and stopped at the given time every day;
3. It is recommended to set **Sensitivity** to the default value of 0.05;

G. Advanced Parameters can significantly affect the results for a particular task, it is recommended to not change those until recommended by Aupera.



Parameter	Value
Debug Mode	Disable
Report Interval (s)	5
Head Confidence Threshold	0.5
Detect Interval (s)	5
Max Keep Alive	15
Min Hits	1
Affinity Threshold	0.008
Max Cluster Size	100
Min Person Area	200

Figure 2.12. Aupera web application page – crowd flow task advanced AI parameters

H. To start the task, click the **Submit** button. After that, a pop-up message will notify you that the task was successfully launched.

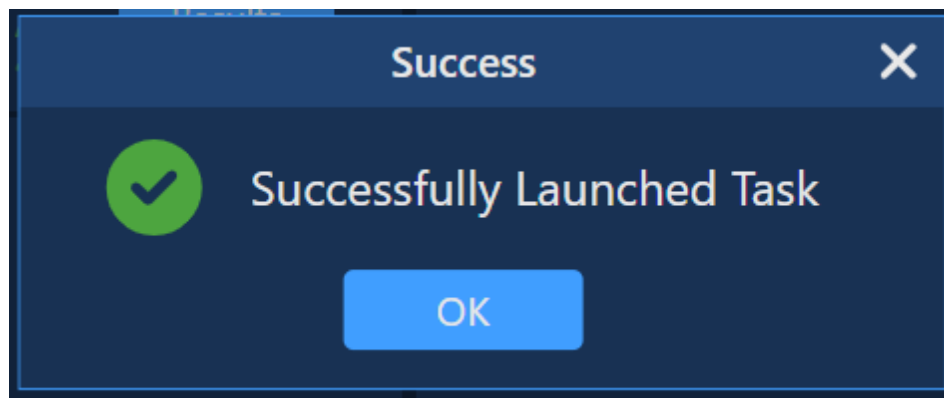


Figure 2.13. Aupera web application page – success notification

I. If the pop-up message reports an error, try launching the task with default parameters or check the settings.

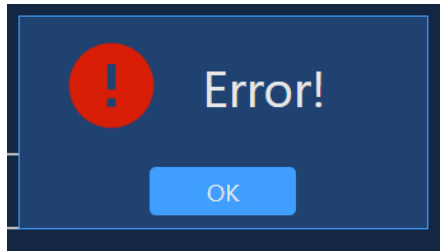


Figure 2.14. Aupera web application page – error notification

J. If the task was launched, the task will appear in the task table on the left side of the screen

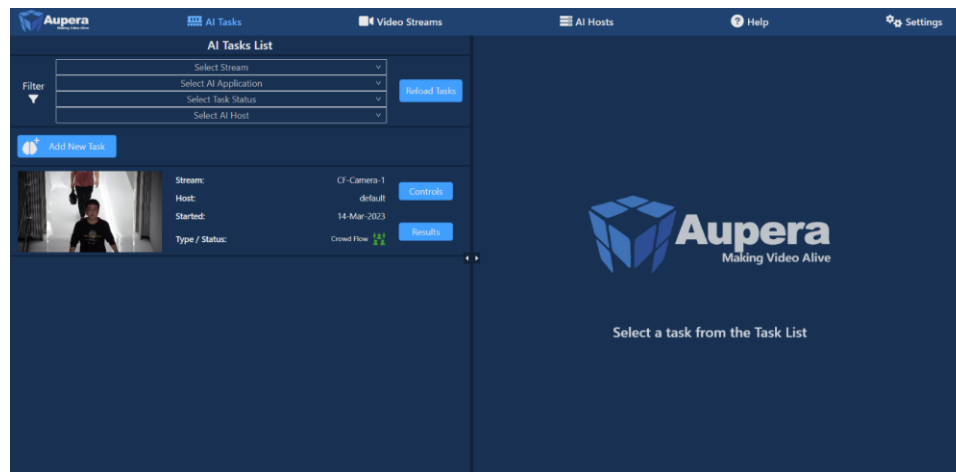


Figure 2.15. Aupera web application page – crowd flow task control

K. Select a task in task list or click **control** button on the row of the task to show the Crowd Flow Control on the right side of the screen.

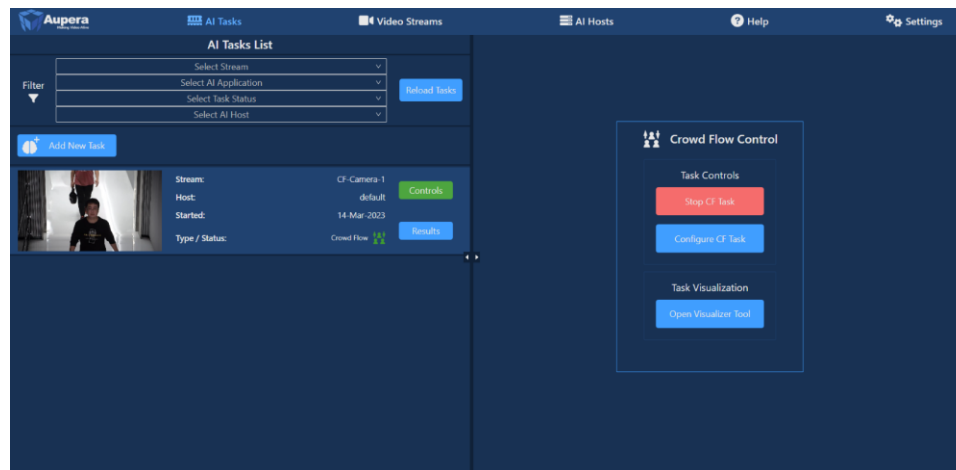


Figure 2.16. Aupera web application page – crowd flow task control

L. To view results, click the **result** button on the specific task row on the table. The right panel will load the result of the specific task selected.

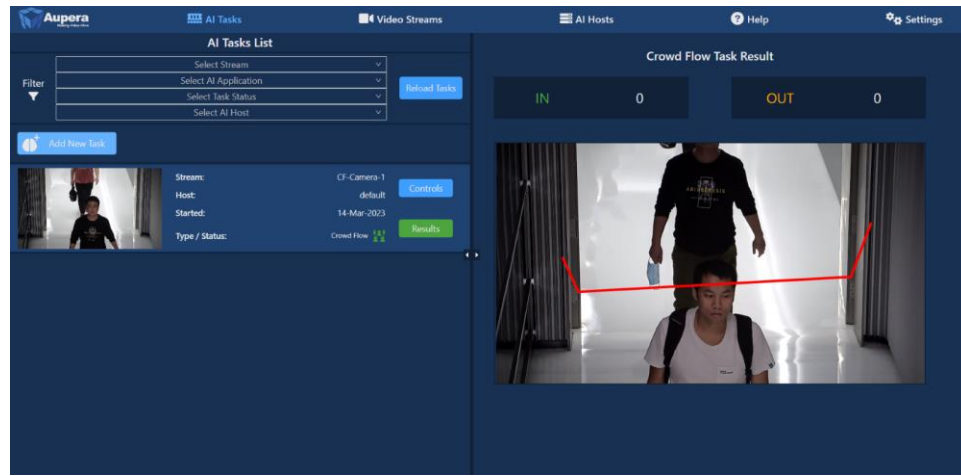


Figure 2.17. Aupera web application page – crowd flow task result

M. To visualize the AI pipeline of the task, click the **Open Visualizer Tool** Button at the bottom of the controls.

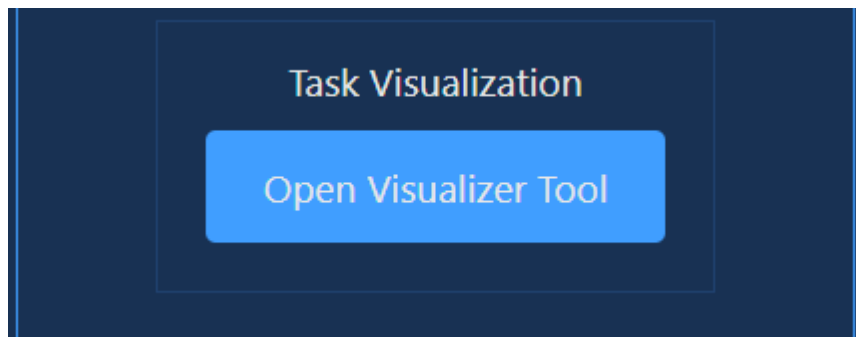
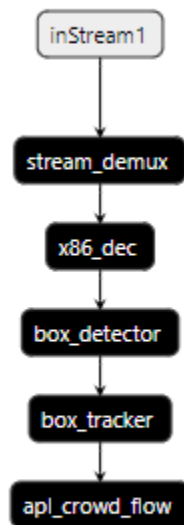


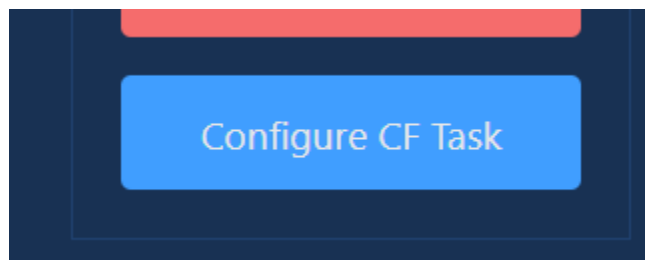
Figure 2.18. Aupera web application page – crowd flow task visualizer tool

The Visualizer will open in a new tab. Then, the AI pipeline graph representing current Crowd Flowd task will appear.



**Figure 2.19. Aupera web application page – crowd flow task AI pipeline graph**

N. To update Crowd Flow task set up, click **Configure CF Task** button, then the Crowd Flow Task Setup window will appear. Any parameter of the running task can be changed. See Figure 2.11 to see more details in Crowd Flow Task set up. After completing the changes to Crowd Flow Task setup, click **Update** button to save changes. Updating task does not reset result counters.



**Figure 2.20. Aupera web application page – crowd flow Configure Crowd Flow Task Button**

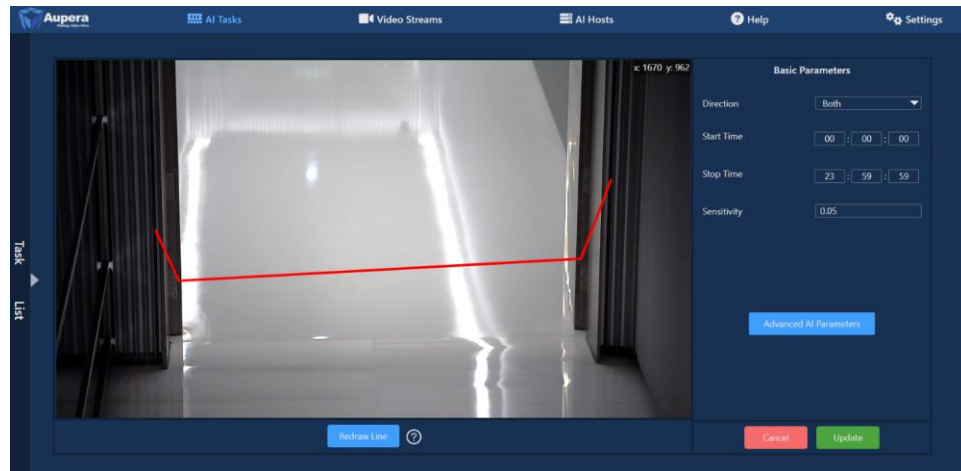


Figure 2.21. Aupera web application page – Configure Crowd Flow Task popup

O. To stop the task, click **Stop CF Task** button, then Stop Crowd Flow Task confirmation will notify that the current task will be stopped. To proceed deleting the task, click **OK** button. To undo stop task, click **Cancel** button.

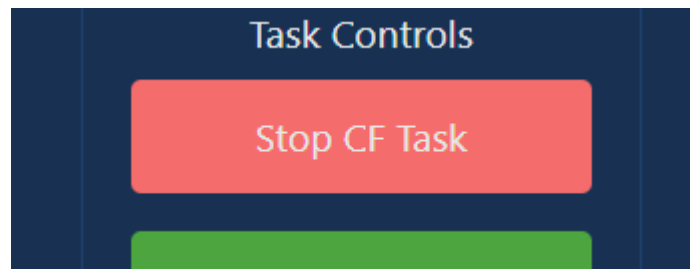


Figure 2.22. Aupera web application page – crowd flow Stop Crowd Flow Task Button

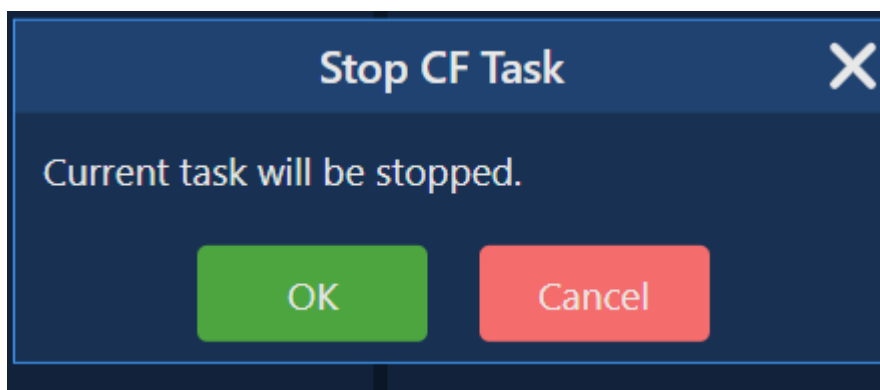


Figure 2.23. Aupera web application page – crowd flow Stop Crowd Flow Task confirmation popup



## 2.4 Building Your Own (Custom) Pipelines

A. Before running a custom pipeline task, to build a custom pipeline, click **Add New Task** button, select a video stream from the drop down, and in **Select Application Type** drop down, select Custom Pipeline

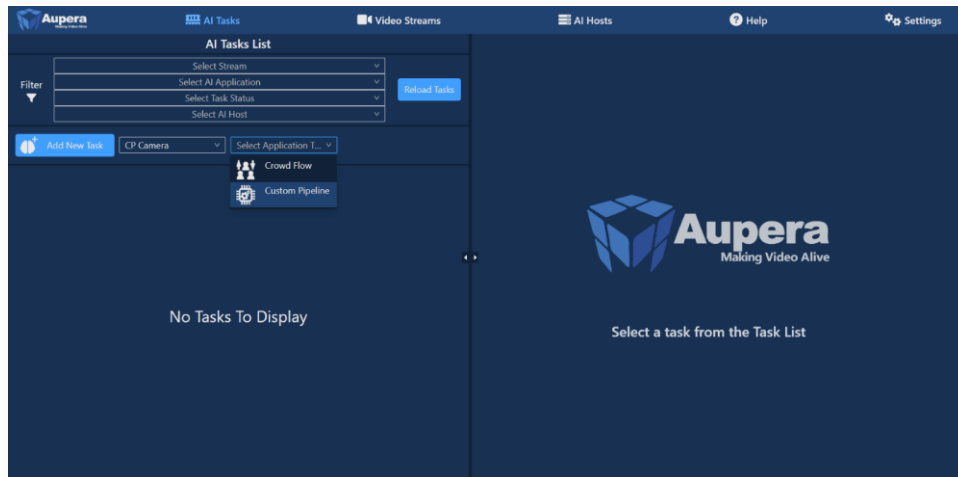


Figure 2.24 Aupera web application page – AI Tasks Hub Page, Video Stream Selected

B. After Stream, Application (Custom Pipeline) and AI Hosts are selected, Custom Pipeline AI Editor will appear. (This is the right panel of the AI Tasks page extended, collapsing this window by clicking **Task List** or **cancel** button will return to the previous page) In the Editor you can type a PBTEXT configuration or **Import From File** it from a file by clicking the corresponding button. You could also click **Download PBTXT** to save content of the editor.

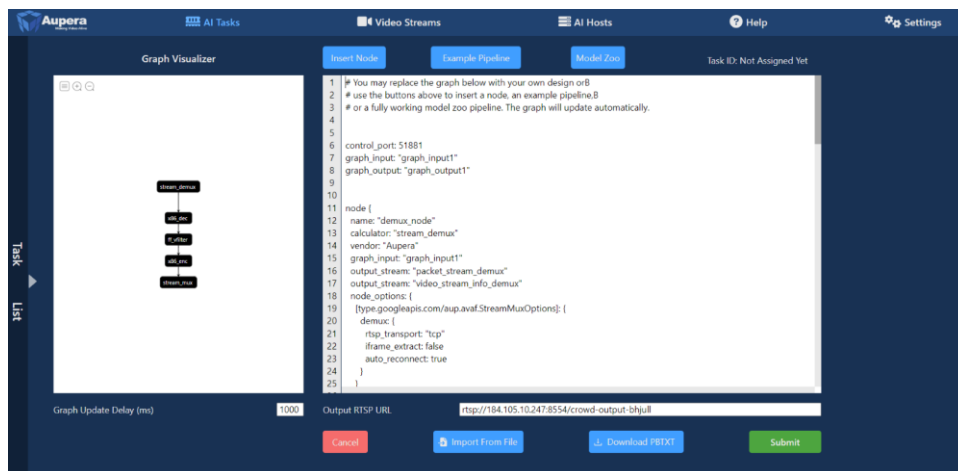


Figure 2.25. Aupera web application page – custom pipeline task AI editor

C. You can also build a pipeline from scratch by using the node tool kit. To do that, insert a node from **Insert Node** button. Click on the input field where you want the node to be inserted, then select a node to insert from the dropdown.

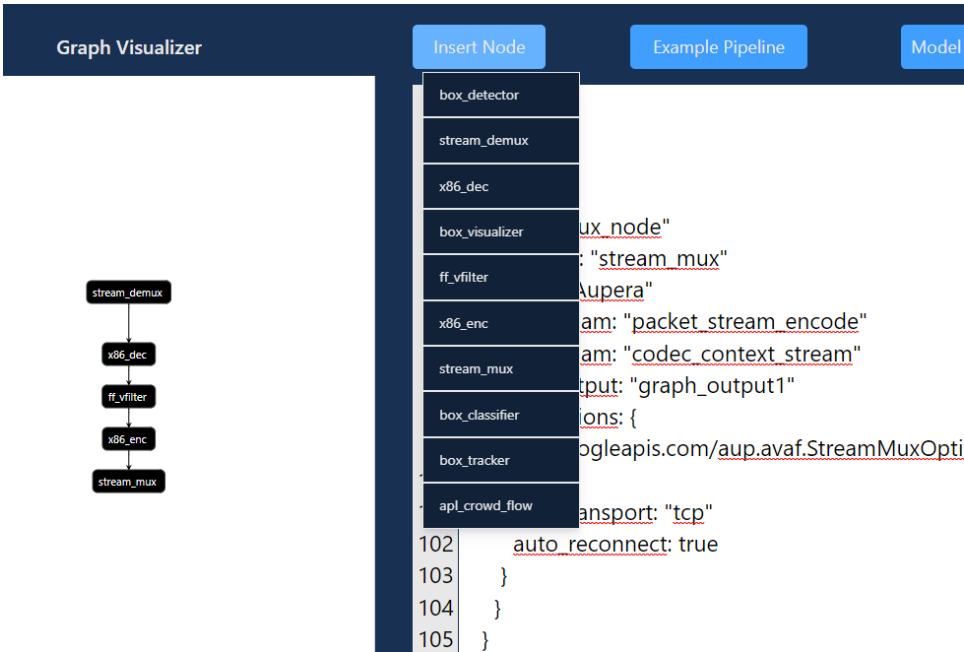


Figure 2.26. Aupera web application page – custom pipeline task AI editor Insert Node selected

D. You can also try Example Pipelines from **Example Pipeline** button. When an Example Pipeline is selected from the dropdown, the Input will be entirely replaced with the example pipeline



Figure 2.27. Aupera web application page – custom pipeline task AI editor Example Pipeline selected

## 2.5 Model Zoo Evaluation

A. Aside from building your own custom pipeline, you can also use our Model Zoo feature. Select **Model Zoo** button to look at the available models. Models can be filtered by Vendor, Category, and Type.

Application	Model Name	Category	Vendor	Hardware Engine	Hardware Type	Float Accuracy	Quantized Accuracy	Input Bits	OPS
person-head	cf_ssd-resnet18-person-head-1000x560x560_2.0	detection	Aupera	OpenCL	xilinx_vck5000_gemini16_nidma_base_1	0.8037	0.6321	160x560	6.30
refnet	dk_yolo2_voc_416_416_5.460_2.0	detection	Xilinx	OpenCL	xilinx_vck5000_gemini16_nidma_base_1	0.9739	0.9650	416x416	5.460
general	dk_yolo2_voc_448_448_5.440_2.0	detection	Xilinx	OpenCL	xilinx_vck5000_gemini16_nidma_base_1	0.7645	0.7740	448x448	342
general	dk_yolo2_voc_448_448_5.666_11.560_2.0	detection	Xilinx	OpenCL	xilinx_vck5000_gemini16_nidma_base_1	0.7700	0.7610	448x448	11.560
general	dk_yolo2_voc_448_448_5.711_5.690_2.0	detection	Xilinx	OpenCL	xilinx_vck5000_gemini16_nidma_base_1	0.7670	0.7540	448x448	5.690
general	dk_yolo2_voc_448_448_5.777_7.830_2.0	detection	Xilinx	OpenCL	xilinx_vck5000_gemini16_nidma_base_1	0.7576	0.7500	448x448	7.820
general	dk_yolo2_voc_416_416_5.420_2.0	detection	Xilinx	OpenCL	xilinx_vck5000_gemini16_nidma_base_1	0.8340	0.8130	416x416	5.430
refnet	dk_yolo2_voc_416_416_5.420_2.0	detection	Xilinx	OpenCL	xilinx_vck5000_gemini16_nidma_base_1	0.9058	0.4910	288x512	53.70
refnet	dk_yolo2_voc_416_416_5.460_2.0	detection	Xilinx	OpenCL	xilinx_vck5000_gemini16_nidma_base_1	0.9520	0.5300	256x512	5.460
general	cf_yolo2_voc_416_416_5.630_2.0	detection	Xilinx	OpenCL	xilinx_vck5000_gemini16_nidma_base_1	0.7846	0.7729	416x416	5.630
general	cf_refnetdet_coco_360_360_120_2.0	detection	Xilinx	OpenCL	xilinx_vck5000_gemini16_nidma_base_1	0.6028	0.7042	360x360	120
general	cf_refnetdet_coco_360_360_0.8_250_2.0	detection	Xilinx	OpenCL	xilinx_vck5000_gemini16_nidma_base_1	0.6794	0.6790	360x360	250
general	cf_refnetdet_coco_360_360_0.92_10.190_2.0	detection	Xilinx	OpenCL	xilinx_vck5000_gemini16_nidma_base_1	0.5409	0.6407	360x360	10.190
general	cf_refnetdet_coco_360_360_0.96_5.080_2.0	detection	Xilinx	OpenCL	xilinx_vck5000_gemini16_nidma_base_1	0.6120	0.6112	360x360	5.080
general	cf_refnetdet_VOC_320_320_81.90_2.0	detection	Xilinx	OpenCL	xilinx_vck5000_gemini16_nidma_base_1	0.8015	0.7999	320x320	81.90
refnet	cf_refnetdet_coco_360_360_0.99_5.30_2.0	detection	Xilinx	OpenCL	xilinx_vck5000_gemini16_nidma_base_1	0.4207	0.4200	360x360	5.30

Figure 2.28. Aupera web application page – custom pipeline task AI editor Model Zoo Selected

Application	Model Name	Category	Vendor	Hardware Engine	Hardware Type	Float Accuracy	Quantized Accuracy	Input Size	OPS
person-head	cf_SSD-RESNET18_PERSON-HEAD-SHIUN_960_540_0.63G_2.0	detection	Aupera	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.8037	0.6323	960x540	6.3G

Figure 2.29. Aupera web application page – custom pipeline task AI editor Model Zoo Selector, filter by Vendor “Aupera”

Application	Model Name	Category	Vendor	Hardware Engine	Hardware Type	Float Accuracy	Quantized Accuracy	Input Size	OPS
general	cf_resnet18_imagenet_224_224_3.65G_2.0	classification	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.6844/0.8864	0.6699/0.8826	224*224	3.65G
general	cf_resnet50_imagenet_224_224_7.7G_2.0	classification	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.7444/0.9185	0.7335/0.9130	224*224	7.7G
general	cf_inceptionv1_imagenet_224_224_3.16G_2.0	classification	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.7030/0.8971	0.6964/0.8942	224*224	3.16G
general	pt_resnet50_imagenet_224_224_4.1G_2.0	classification	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.7610/0.9290	0.7603/0.9280	224*224	4.1G
general	tf2_resnet50_imagenet_224_224_7.76G_2.0	classification	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.7513/0.9220	0.7437/0.9214	224*224	7.76G
general	tf_inceptionv1_imagenet_224_224_3G_2.0	classification	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.6976/0.8963	0.6786/0.8852	224*224	3.0G
general	tf_milperf_resnet50_imagenet_224_224_8.19G_2.0	classification	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.7652/0.9301	0.7536/0.9264	224*224	8.19G
general	tf_resnetv1_101_imagenet_224_224_14.4G_2.0	classification	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.7640/0.9289	0.7414/0.9176	224*224	14.4G
general	tf_resnetv1_152_imagenet_224_224_21.83G_2.0	classification	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.7681/0.9317	0.7464/0.9220	224*224	21.83G
general	tf_resnetv1_50_imagenet_224_224_6.97G_2.0	classification	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.7520/0.9219	0.7411/0.9167	224*224	6.97G

Figure 2.30. Aupera web application page – custom pipeline task AI editor Model Zoo Selector, filter by Category “classification”

Model Zoo Selector									
Select Vendor		Select Category		xilinx_vck5000_gen3x16_xdma_base_1					
Application	Model Name	Category	Vendor	Hardware Engine	Hardware Type	Accuracy	Accuracy	Input Size	OPS
person-head	cf_SSD-RESNET18_PERSON-HEAD-SHIUN_960_540_0.63G_2.0	detection	Aupera	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.8037	0.6323	960x540	6.3G
retail	dk_tiny-yolov3_416_416_5.46G_2.0	detection	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.9739	0.9650	416x416	5.46G
general	dk_yolov2_voc_448_448_34G_2.0	detection	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.7845	0.7740	448x448	34G
general	dk_yolov2_voc_448_448_0.66_11.56G_2.0	detection	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.7700	0.7610	448x448	11.56G
general	dk_yolov2_voc_448_448_0.71_9.86G_2.0	detection	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.7670	0.7540	448x448	9.86G
general	dk_yolov2_voc_448_448_0.77_7.82G_2.0	detection	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.7576	0.7500	448x448	7.82G
general	dk_yolov3_voc_416_416_65.42G_2.0	detection	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.8240	0.8130	416x416	65.42G
adas	dk_yolov3_bdd_288_512_53.7G_2.0	detection	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.5058	0.4910	288x512	53.7G
adas	dk_yolov3_cityscapes_256_512_0.9_5.46G_2.0	detection	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.5520	0.5300	256x512	5.46G
general	tf_yolov3_voc_416_416_65.63G_2.0	detection	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.7846	0.7729	416x416	65.63G
general	cf_refinedet_coco_360_480_123G_2.0	detection	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.6928	0.7042	360x480	123G
general	cf_refinedet_coco_360_480_0.8_25G_2.0	detection	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.6794	0.6790	360x480	25G
general	cf_refinedet_coco_360_480_0.92_10.10G_2.0	detection	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.6489	0.6487	360x480	10.10G
general	cf_refinedet_coco_360_480_0.96_5.08G_2.0	detection	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.6120	0.6112	360x480	5.08G
general	tf_refinedet_VOC_320_320_81.9G_2.0	detection	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.8015	0.7999	320x320	81.9G
adas	cf_ssadas_bdd_360_480_0.95_6.3G_2.0	detection	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1	0.4207	0.4200	360x480	6.3G

Figure 2.31. Aupera web application page – custom pipeline task AI editor Model Zoo Selector, filter by hardware type..

B. To insert the wanted model, select a model, then choose Pipeline PBTXT or Throughput PBTXT (if available). After choosing a model that is compatible with the AI host selected (for more information about AI host, go to 2.8 AI Hosts Page), the Model Zoo Selector popup will automatically close, and the input will be entirely replaced with the model pbtxt selected.

Model Zoo Selector					
Select Vendor		Select Category			
Application	Model Name	Category	Vendor	Hardware Engine	Hardware Type
person-head	cf_SSD-RESNET18_PERSON-HEAD-SHIUN_960_540_0.63G_2.0	detection	Aupera	8pe	xilinx_vck5000_gen3x16_xdma_base_1
retail	dk_tiny-yolov3_416_416_5.46G_2.0	detection	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1
general	dk_yolov2_voc_448_448_34G_2.0	detection	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1
general	dk_yolov2_voc_448_448_0.66_11.56G_2.0	Pipeline PBTXT Throughput PBTXT		8pe	xilinx_vck5000_gen3x16_xdma_base_1
general	dk_yolov2_voc_448_448_0.71_9.86G_2.0			8pe	xilinx_vck5000_gen3x16_xdma_base_1
general	dk_yolov2_voc_448_448_0.77_7.82G_2.0	detection	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1
general	dk_yolov3_voc_416_416_65.42G_2.0	detection	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1
adas	dk_yolov3_bdd_288_512_53.7G_2.0	detection	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1
adas	dk_yolov3_cityscapes_256_512_0.9_5.46G_2.0	detection	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1
general	tf_yolov3_voc_416_416_65.63G_2.0	detection	Xilinx	8pe	xilinx_vck5000_gen3x16_xdma_base_1

**Figure 2.32. Aupera web application page – custom pipeline task AI editor Model Zoo Selector, model selected**

The option **Throughput PBTXT** will run on images. It loads images that are pre-resized to match the machine learning network input size requirements into memory and loops over them. The output result of throughput PBTXT will just contain JSON/text notification.

The option **Pipeline PBTXT** will also contain transcoding or visualization.

For more information about the results of the custom pipeline, refer to the next section (Point G.).

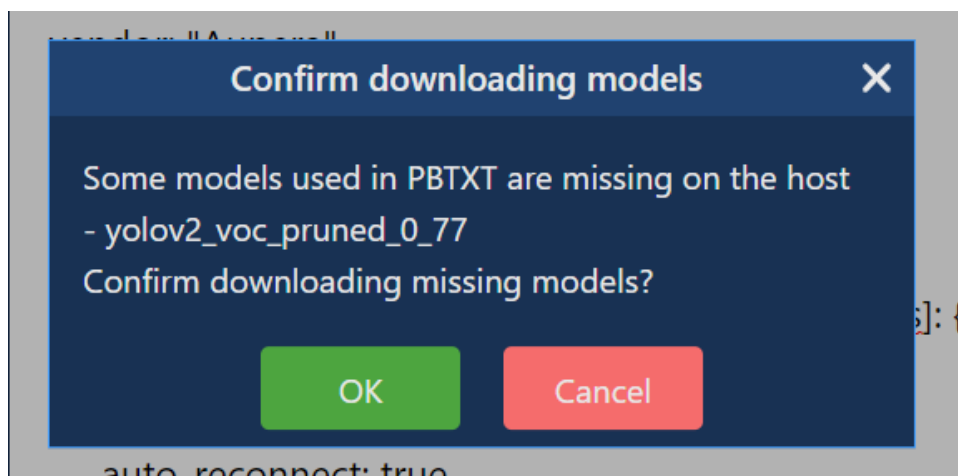
**NOTE:** Before using Insert Node/Example Pipelines/Model Zoo, make sure that the machine has internet access.

## 2.6 Running Your Own (Custom) Pipeline

A. After completing building the custom pipeline either from scratch or model zoo, you can start running the pipeline. Before submitting the task, please enter *Output RTSP URL* (this value needs to follow a specific format of **rtsp://<vmaccel\_instance\_ip\_address>:8554/<user\_specified\_name>**). Click Submit to start the Custom Pipeline task.

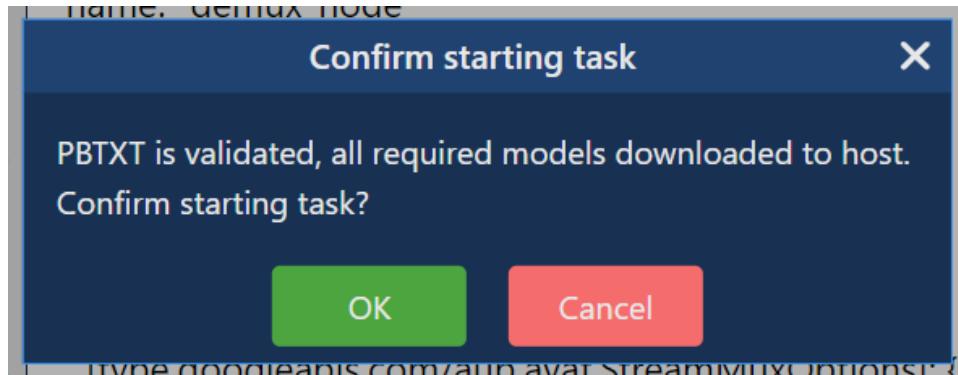
**NOTE:** **<user\_specified\_name>** can be any arbitrary name that the user chooses.

B. After submitting the task, if there are missing models, there will be a popup displayed informing which model is missing. Click **OK** to download model, or **Cancel** to download later.



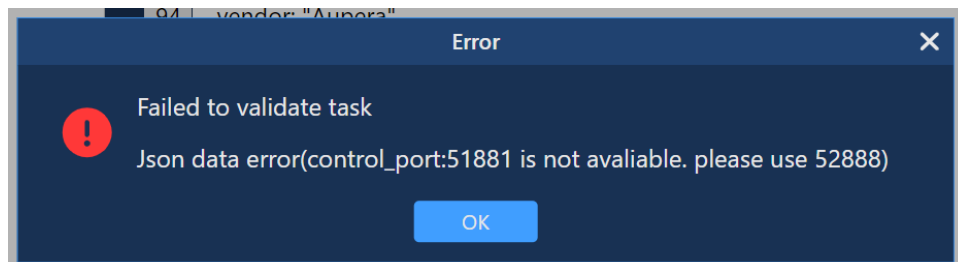
**Figure 2.33. Aupera web application page – Custom Pipeline Submit Task download model popup.**

Or, if all models are validated on current AI host, the popup will give the option to start custom pipeline task immediately. Click **OK** to start task or **Cancel** to start later.



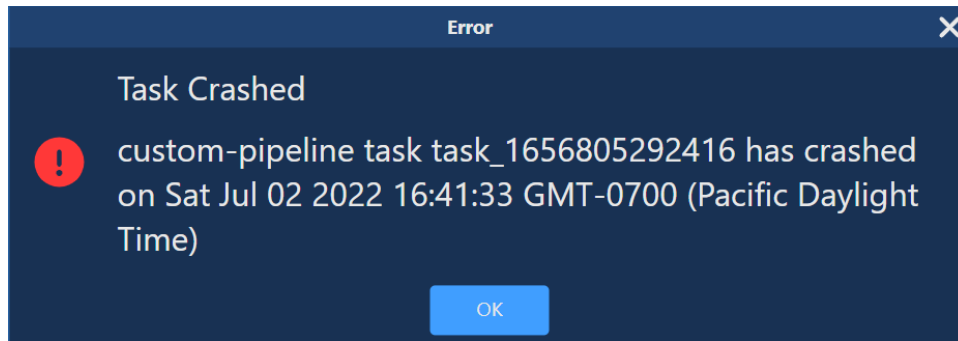
**Figure 2.34. Aupera web application page – Custom Pipeline Submit Task confirm starting task popup.**

If Task cannot run due to unavailable control port, The following popup will show. In this case, click **OK** to close popup, go to editor and change the input control port to an available control port and **Submit** the task again.



**Figure 2.35. Aupera web application page – task launch failed notification**

C. If the task was started, but then crashed after some time, message about that will be displayed



**Figure 2.36. Aupera web application page – task crashed notification**

D. When the task is submitted, it will appear in the task list. Depending on whether the custom pipeline task's current status, status will show different icon.

**Table 1.1. Aupera web application page – Custom pipeline tasks' status icon**

Icons	Definition	Instructions
	Model missing	To download missing model, select entry from table, click <b>Open AI Editor</b> button. Click <b>OK</b> on popup notification on downloading model.
	Model downloading	Wait until model are successfully downloaded to run task.
	Model ready	Model are downloaded and task is ready to run. Select entry from table, click <b>Open AI Editor</b> button. Click <b>OK</b> on popup notification on starting task.
	Task is running	Task is successfully running.
	Task failed	Task failed. Select entry from table, click <b>Open AI Editor</b> button. Click <b>Stop Task</b> button. You can restart task following start custom pipeline task instruction again.

E. When the task row in task list or the **control** button on the row is clicked, the right side of the screen will show the Custom Pipeline control component.



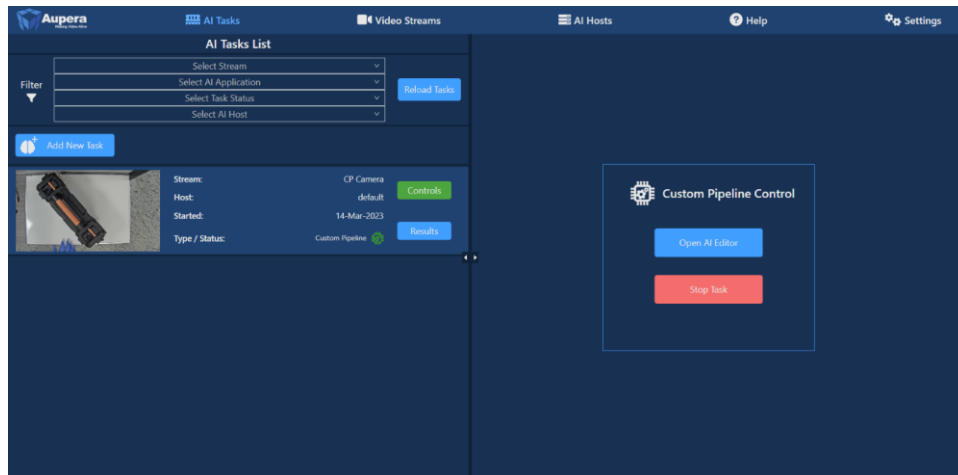


Figure 2.37. Aupera web application page – custom pipeline task control

F. If AI Editor is opened for a running task, current PBTEXT configuration will be displayed in the editor field. However, task update is not supported now, so please stop and start the task again in case any changes in PBTEXT are required.

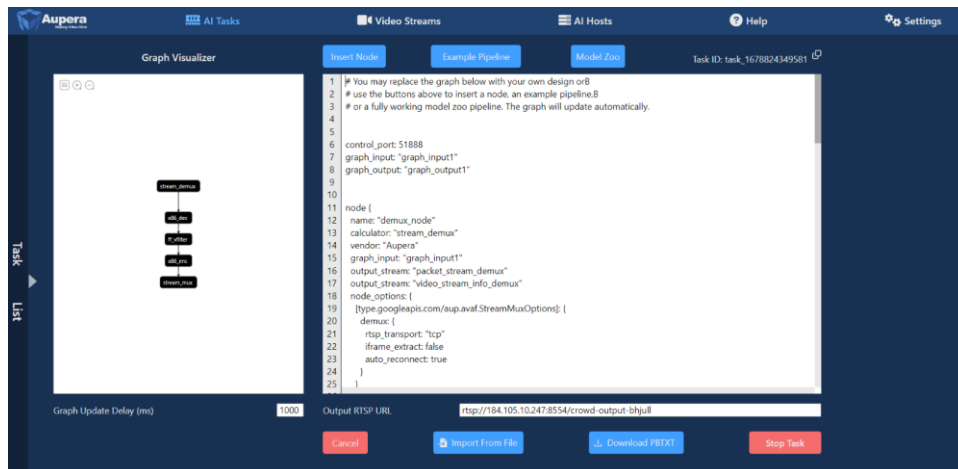


Figure 2.38. Aupera web application page – custom pipeline task pbtxt example

G. To see the Custom Pipeline task results, click **result** button on the specific task in task table.

If you used the model zoo feature and submitted a pipeline pbtxt, both video result output and JSON viewer results will be displayed. If throughput pbtxt is submitted, only JSON viewer results will be displayed.

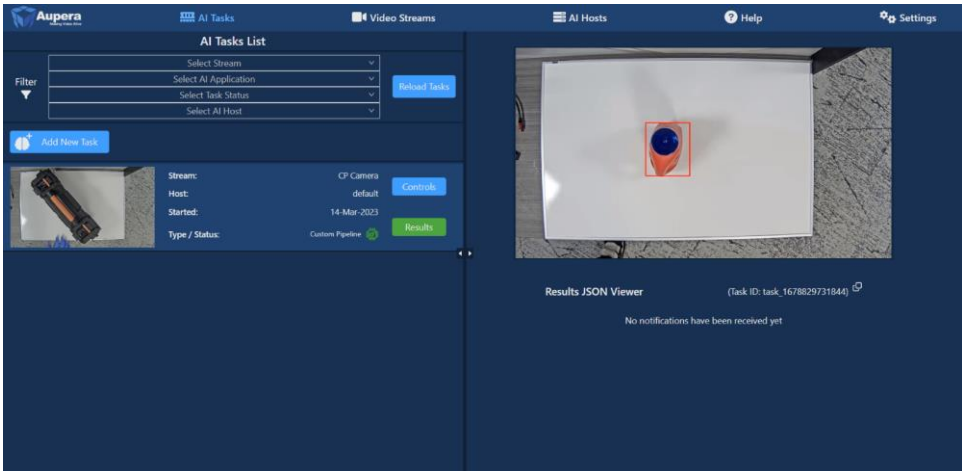


Figure 2.39. Aupera web application page – custom pipeline task result of model zoo pipeline pbt.txt

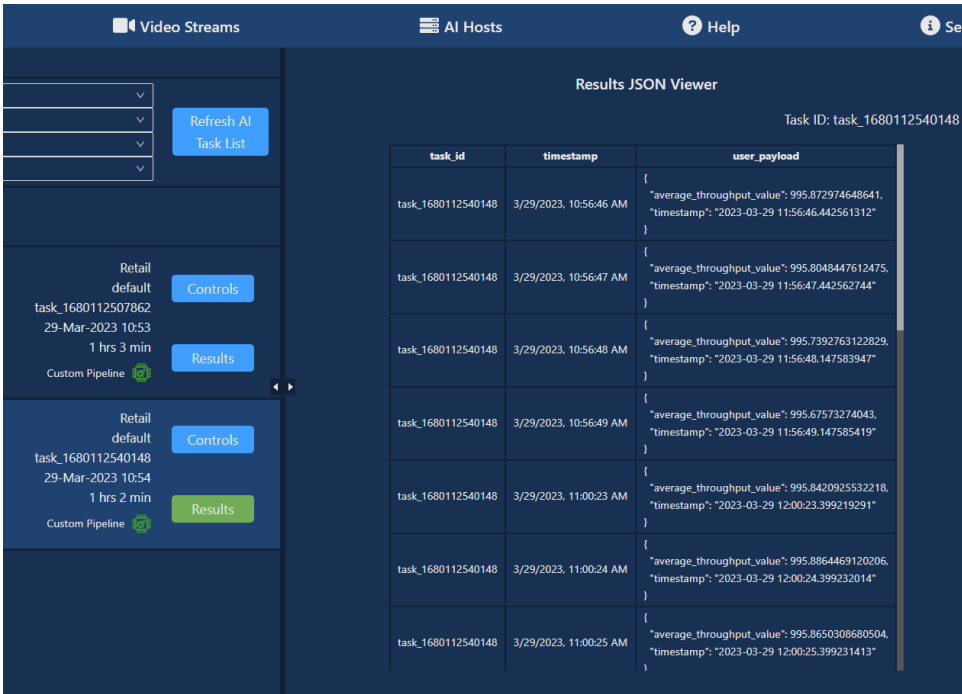


Figure 2.40. Aupera web application page – custom pipeline task result of model zoo throughput pbt.txt

**NOTE:** After the Custom Pipeline Task started, it may take up to a few seconds before an output video stream starts. “Video failed to load” error may occur when checking custom pipeline result right after the task is started. Wait a few seconds and click the refresh button to try displaying the output video again.

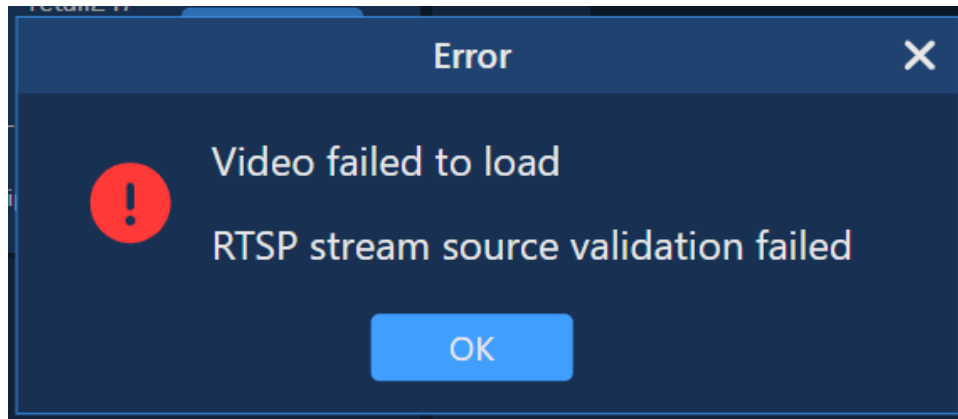


Figure 2.41. Aupera web application page – custom pipeline task result, “Video failed to load” error

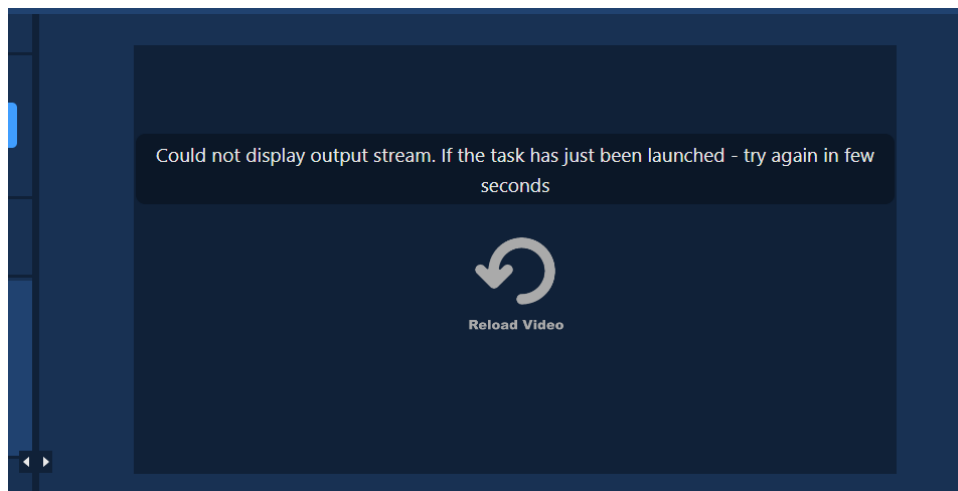


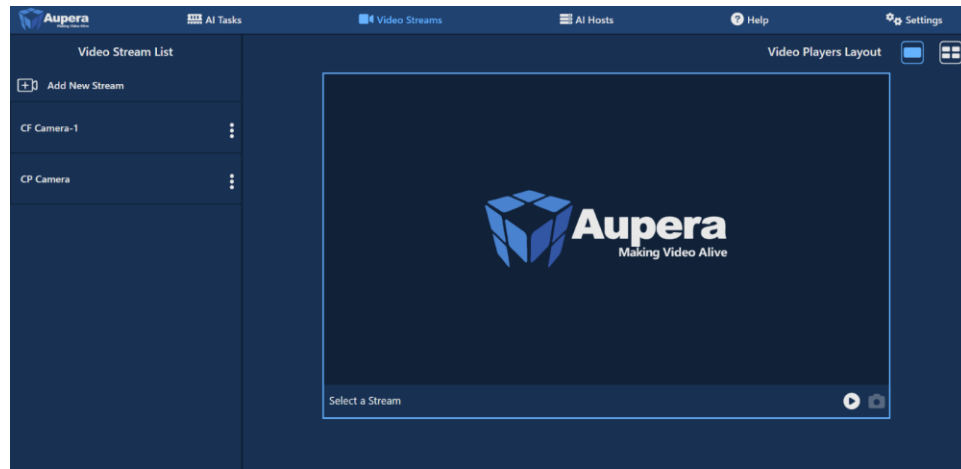
Figure 2.42. Aupera web application page – custom pipeline task result page (output video not yet available)

**NOTE:** Refresh the page if Custom Pipeline task result failed to load after running the task for a long time.

H. To stop the task, click Stop Task in the Custom Pipeline control component.

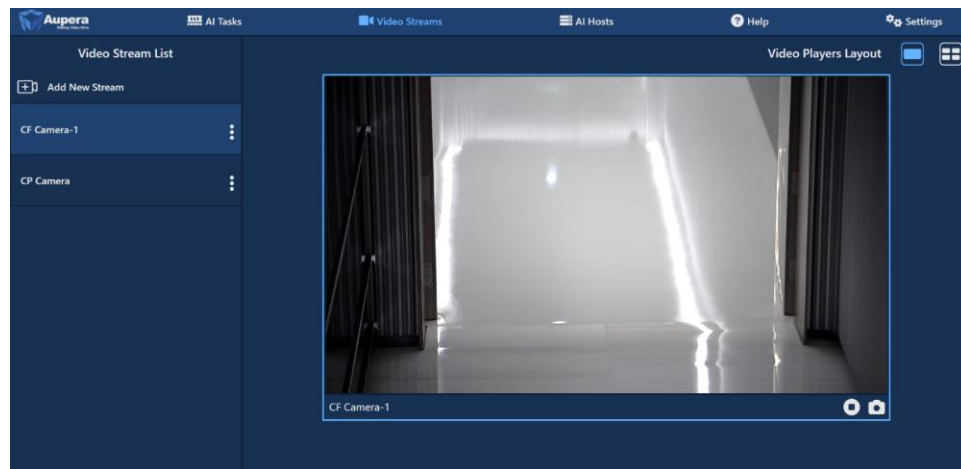
## 2.7 Using Video Streams to Add New Stream, play videos, or change snapshots

A. Click **Video Streams**, then click **Add New Stream** button, then popup will appear shown in Figure 2.4.



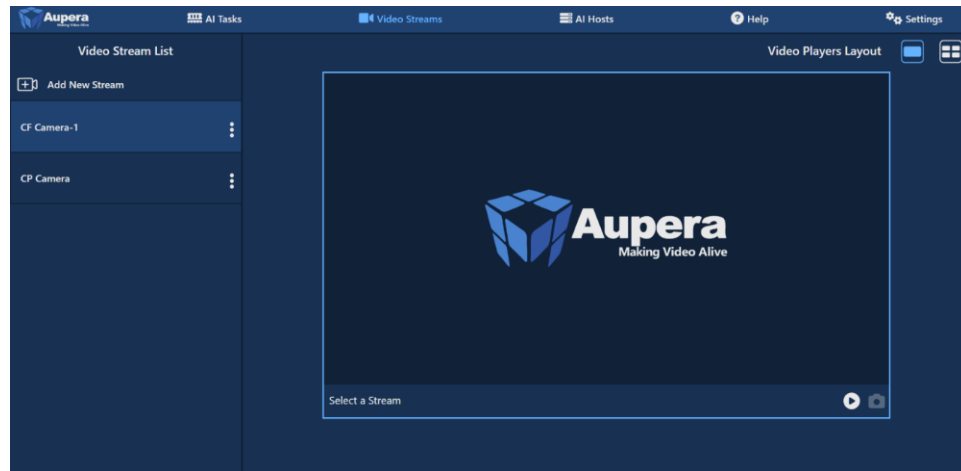
**Figure 2.43 Aupera web application page – Video Streams Page**

B. To play the video, select a Stream from Video Stream List. Video playback will start immediately. If another video stream is clicked, video playback will switch to this video stream.



**Figure 2.44 Aupera web application page – Video Stream Page, video stream selected**

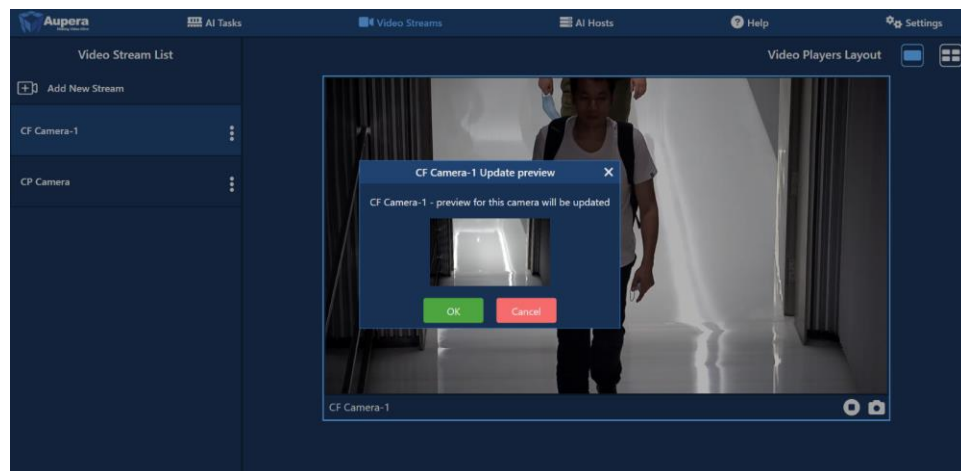
C. The video can also be played and stopped by clicking the stop and play button located in the bottom right corner of the video play.



**Figure 2.45 Aupera web application page – Video Stream Page, stopped button clicked**

**NOTE:** Video player should start playing shortly after the play button is clicked. If the video player does not start playing after clicking the play button for 10 – 15 seconds, click the stop button and play button again to restart the video play.

D. Video stream preview will be taken automatically in several seconds after the stream is played for the first time. To change the snapshot, click the Camera Button beside the play button while the video is playing



**Figure 2.46 Aupera web application page – Video Stream Page, snapshot taken**

Click **OK** button to update the snapshot or **Cancel** button to cancel updating the snapshot.

E. To view four video streams in the same screen, click the four rectangle icon located on the top right corner of the screen. To play video on a specific Video Player, choose a video frame by clicking directly on the video frame and select a

video to play from Video Stream List, or use the Play button at the bottom right corner of each video frame to play or stop video play.

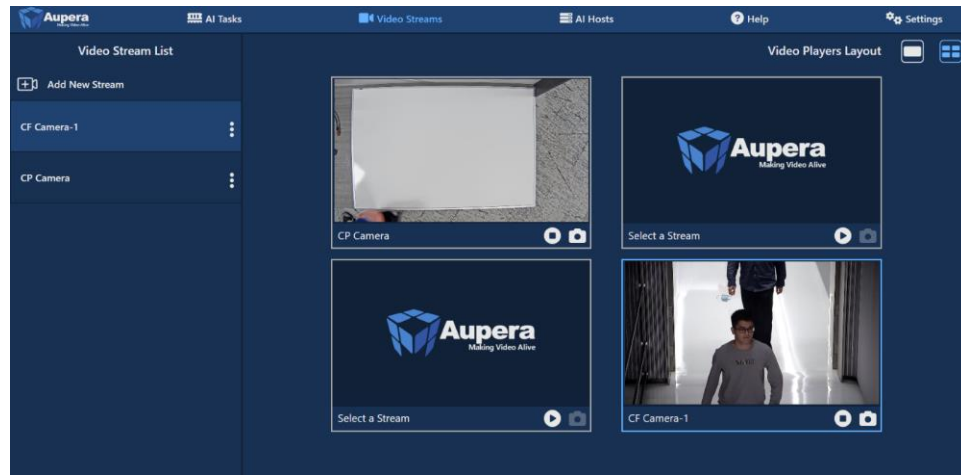


Figure 2.47 Aupera web application page – Video Stream Page, 4 video frames view

## 2.8 AI Hosts Page

A. User will be given a default AI Host when first starting the web application.

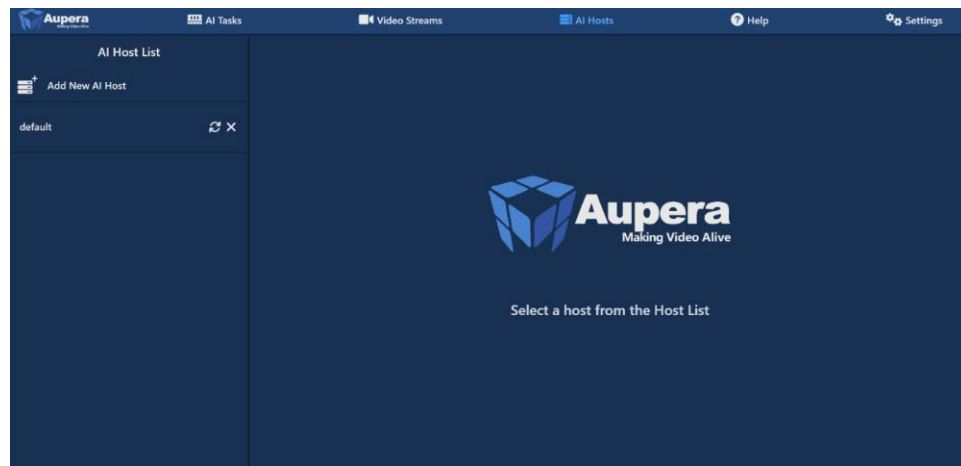
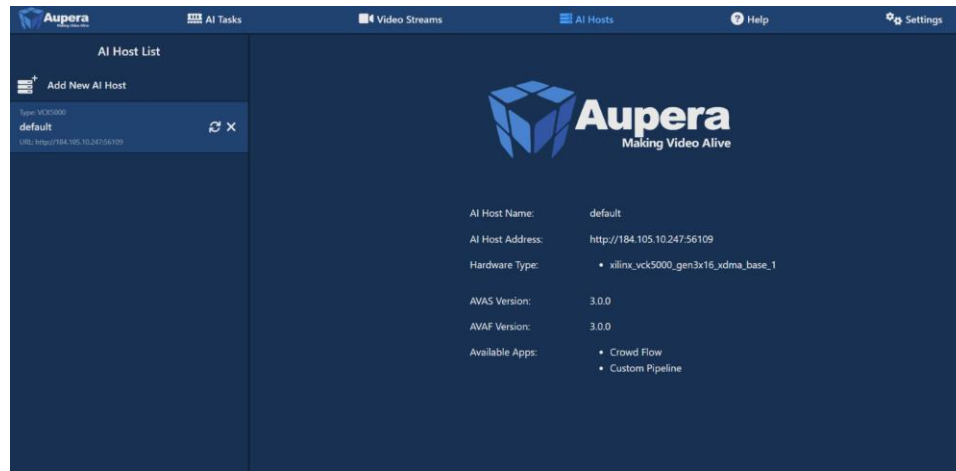


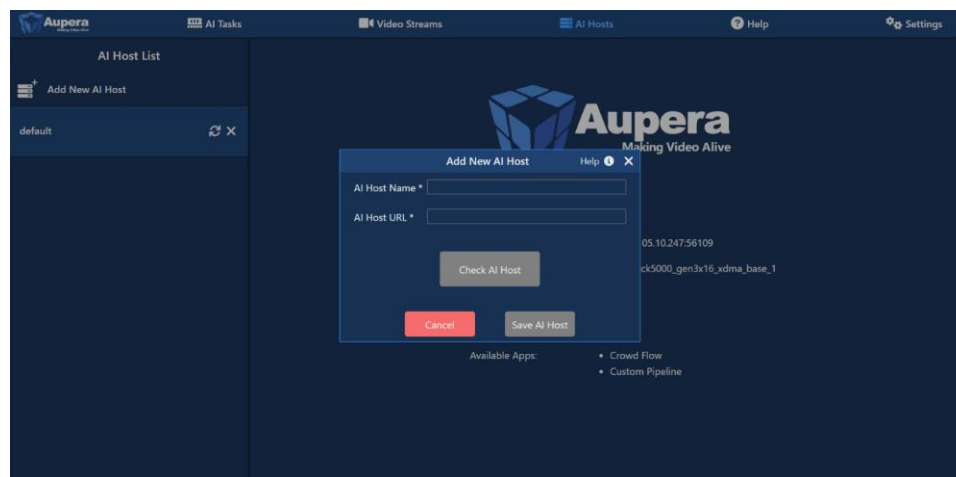
Figure 2.48 Aupera web application page – AI Host Page

B. Information about the AI host can be checked by clicking on the entry of the AI host in AI host list.



**Figure 2.49 Aupera web application page – AI Host Page, AI Host Info**

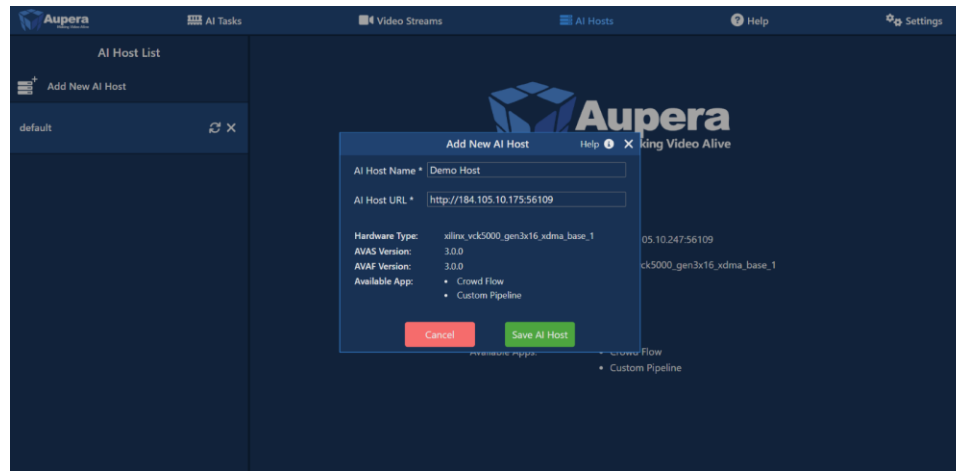
C. To Add a new AI host, click on **Add New AI Host** button and a popup will appear for user to check the AI host url to be added



**Figure 2.50 Aupera web application page – AI Host Page, Add New AI Host**

**NOTE:** If you need help in this step, click **Help** button in Figure 2.49 for more details about each input field and the next steps.

D. Enter a valid AI host name (name cannot be “default”) and URL (AI host URL should start with “http”), then click **Check AI Host** (To successfully save AI host, AI host name and URL must be verified through check AI host information)

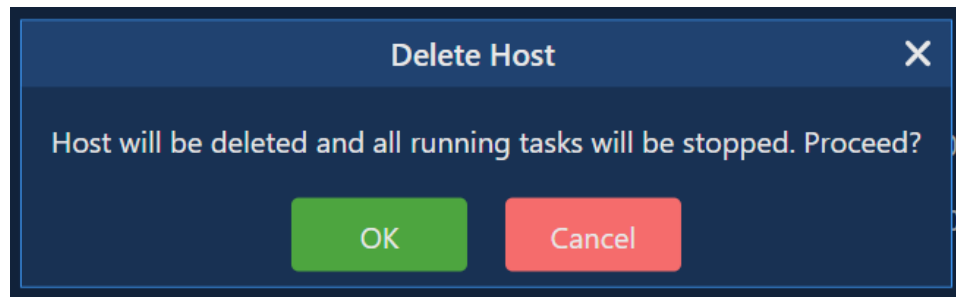


**Figure 2.51 Aupera web application page – Host Page, Add New Host – host URL checked**

During Check AI Host, AI host URL can be modified at any time. Then user can click **Check AI Host** button again if modification of URL is needed.

After clicking the **Submit** button, the new added AI host will appear in the AI Host list and ready to use throughout the entire web application.

E. To delete an AI host, click on the **X** button on the specific AI host that are intended to be deleted on the AI host list. Then select **OK** on the delete AI host confirmation popup



**Figure 2.52 Aupera web application page – Host Page, Delete Host confirmation popup**

**NOTE:** If there is no AI Host added to the web application, the user will not be able to access any task related functions. Therefore, users will not be able to access AI Tasks tab if there is no AI host added. To access and start tasks, make sure there is at least one AI host added, otherwise the AI Tasks tab will be blocked.



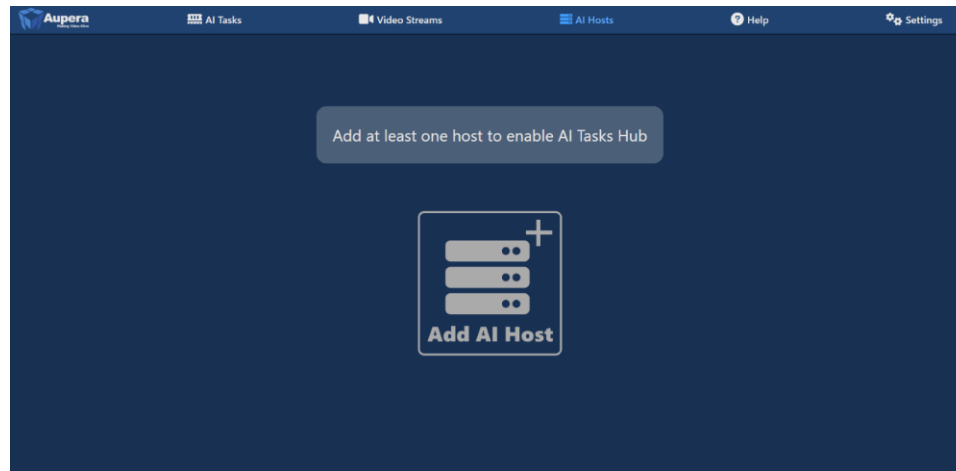


Figure 2.53 Aupera web application page – AI Host Page, No AI Host added

## 2.9 Useful tools

A. **Filter** located at the top of the task list can filter items in task list. To use **Filter**, simply type in the input field or select an item by clicking the input field or the arrow button to filter the task list item by Video Stream, AI Application, Task Status, or AI Hosts

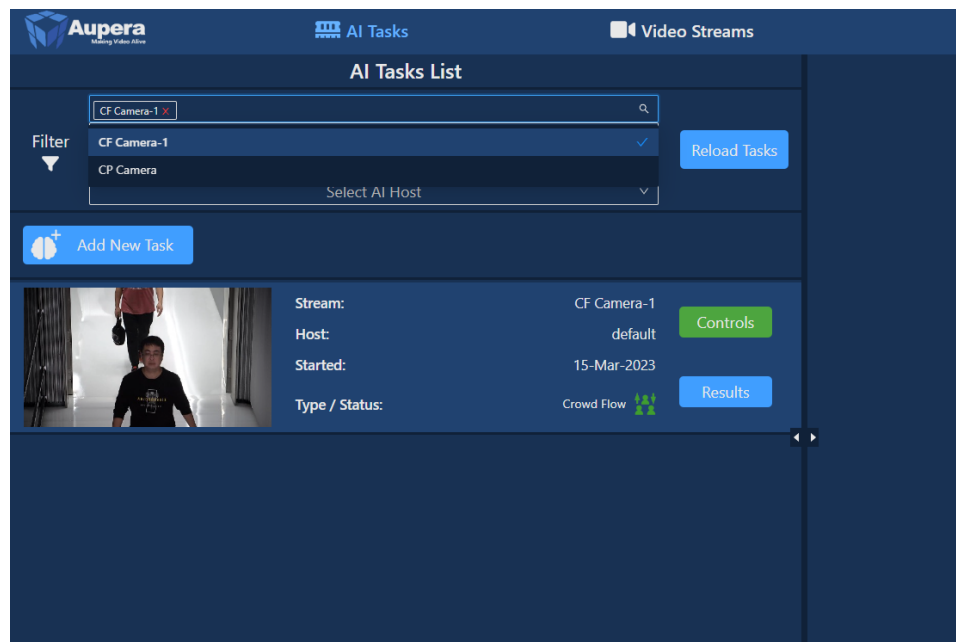


Figure 2.54 Aupera web application page – AI Tasks Hub, filter by Video Stream

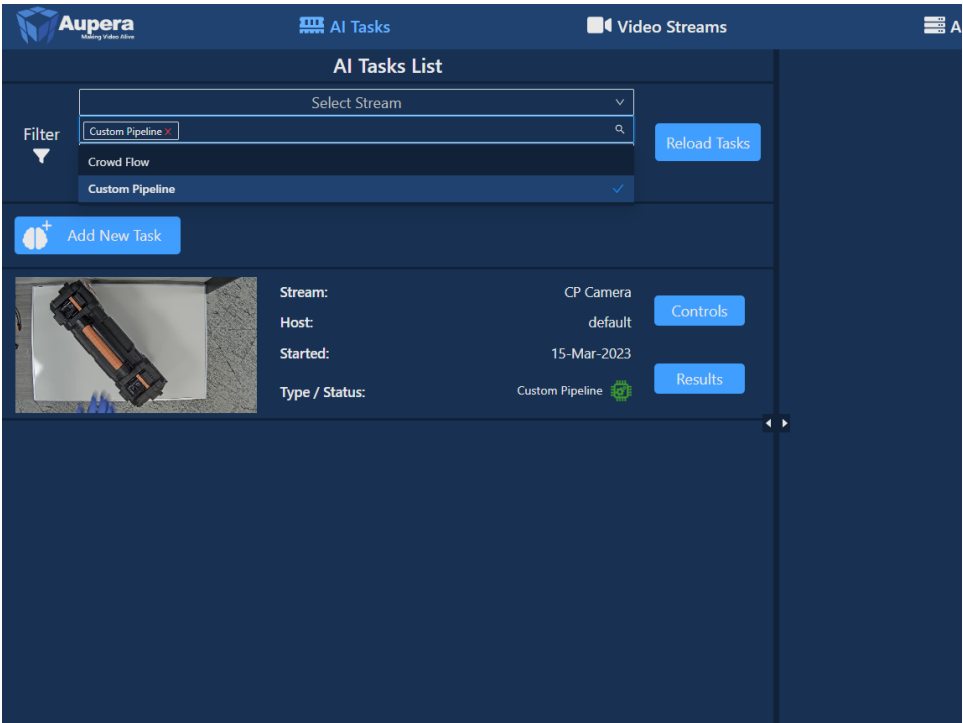


Figure 2.55 Aupera web application page – AI Tasks Hub, filter by AI Application

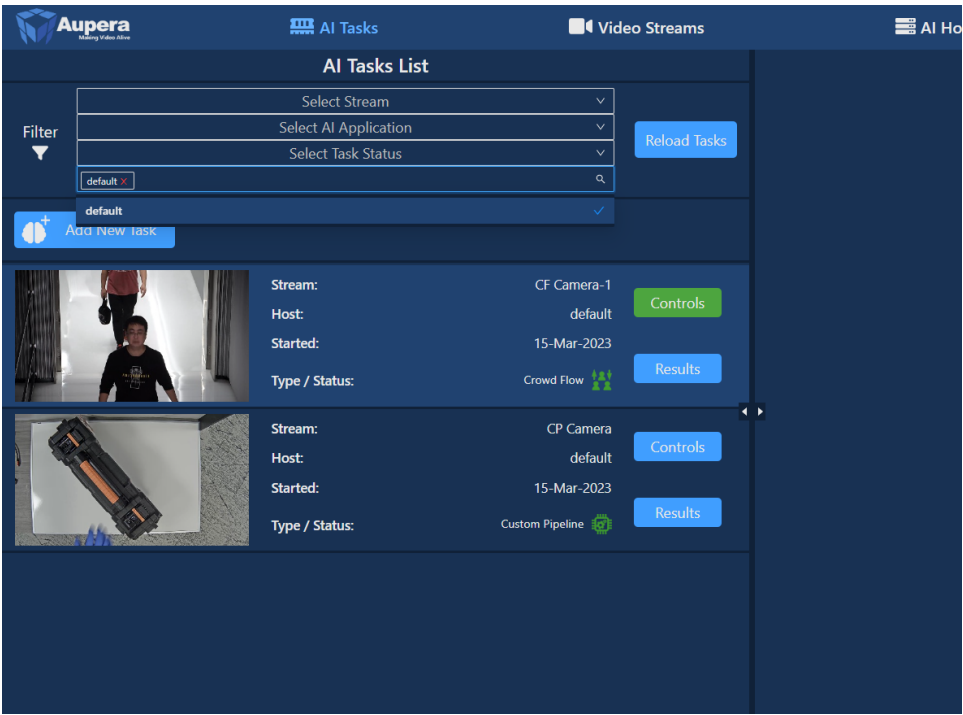


Figure 2.56 Aupera web application page – AI Tasks Hub, filter by AI Host

B. Arrow buttons in the middle of the AI Tasks Hub page help adjust the ratio of the screen for a more convenient and customized view. Right arrow can be clicked to extend the table view. Left arrow can be clicked to extend what exists on the right side of the screen. When one view is extended, a bar will appear to help collapse the screen.

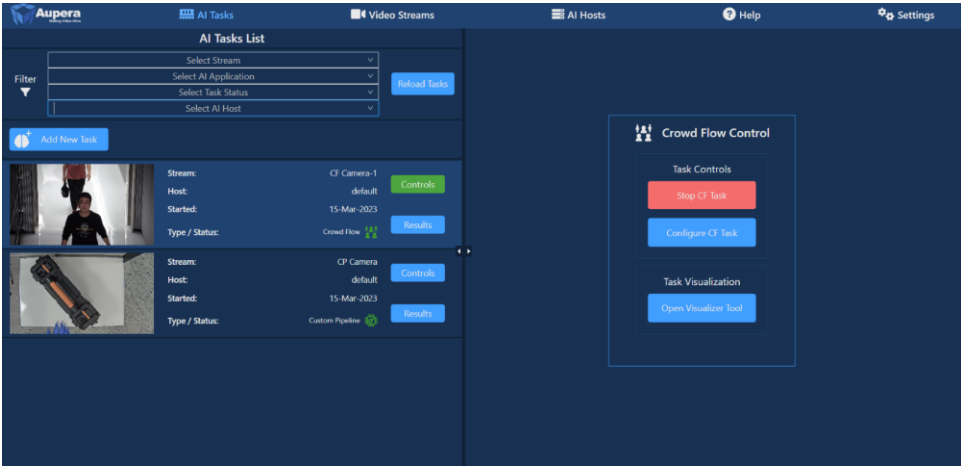


Figure 2.57 Aupera web application page – AI Tasks Hub view on smaller screen

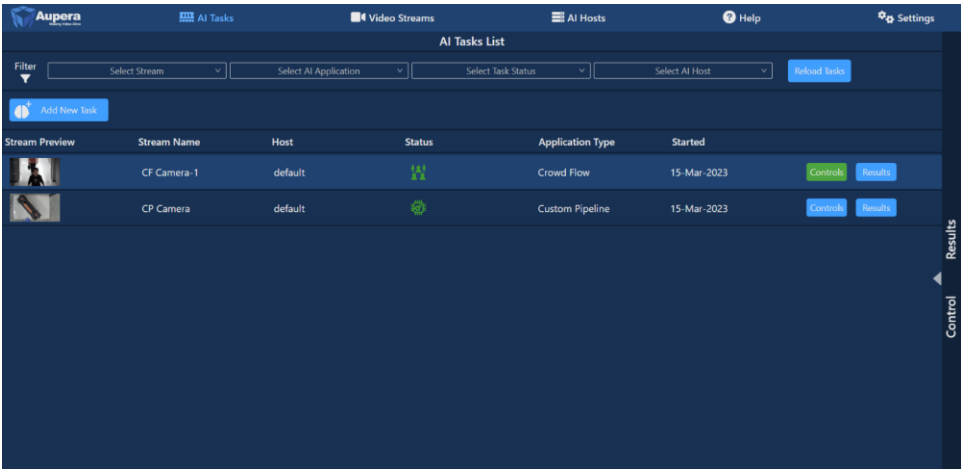


Figure 2.58 Aupera web application page – AI Tasks Hub, table(left) view extended

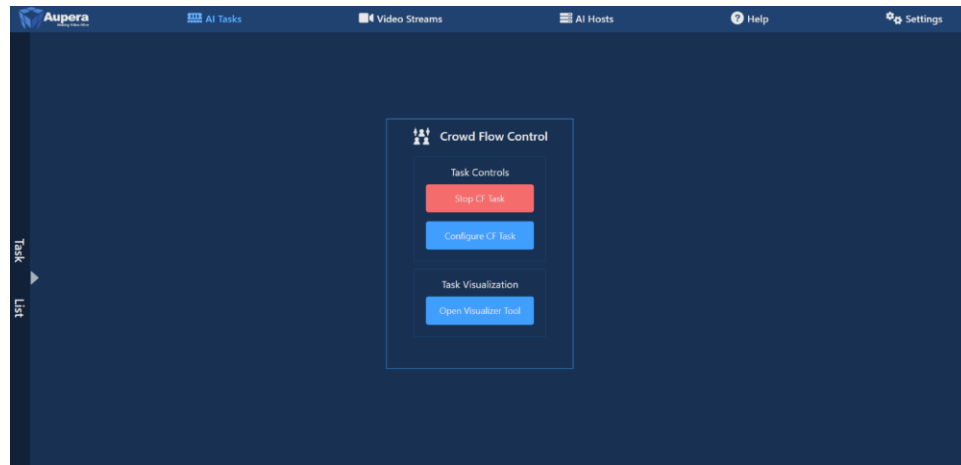


Figure 2.59 Aupera web application page – AI Tasks Hub, right view extended

## 2.10 Launching Your Own RTSP Streams

To test any pipeline, you need either an RTSP stream, your test video, or an RTSP address of your IP camera. If you are using the RTSP streams provided (in NOTE 3 in Section 1.1), you can skip this section and move on to the next section.

This section focuses on helping the user to broadcast their test videos via a RTSP server. Before continuing, please make sure FFMPEG is downloaded on your local machine and accessible via command line. Visit [FFMPEG download](#) page for further instructions.

For your video file residing on your local machine to be pushed into a RTSP server, you can follow the instructions below:

- A. You can publish a stream using

```
ffmpeg -re -stream_loop -1 -i file.ts -c copy -f rtsp  
rtsp://server_ip:8554/mystream
```

Where *file.ts* is your video file residing on your local machine and *server\_ip* is the IP address of your VMAccel instance.

**NOTE:** The key *mystream* (in *rtsp://server\_ip:8554/mystream* above) should be a unique string for each stream.

- B. You can then watch the stream using VLC (or any other video player) through **media/open network** stream option, by hitting *ctrl+n* or by using

```
vlc rtsp://server_ip:8554/mystream
```

You can download VLC from [here](#).

### 3 RUNNING AUPERA VMSS2.0 PIPELINES ON SERVER (VIA COMMAND LINE)

In this section, we will describe how to run the Aupera VMSS2.0 pipelines through the VMSS2.0 Server via the command line. To use AVAS, the VMSS2.0 Server, users need to launch a VMAccel terminal and go into the docker for VMSS2.0 Server. Once there, users can run any pipelines directly from the command line. In what follows, we will introduce the procedures to access AVAS docker, run VMSS2.0 pipelines, and offer some pipeline examples in detail.

#### 3.1 Accessing VMSS2.0 Server Docker

A. In the left-hand sidebar select “**Instances**”. Then, click on the name of your instance.

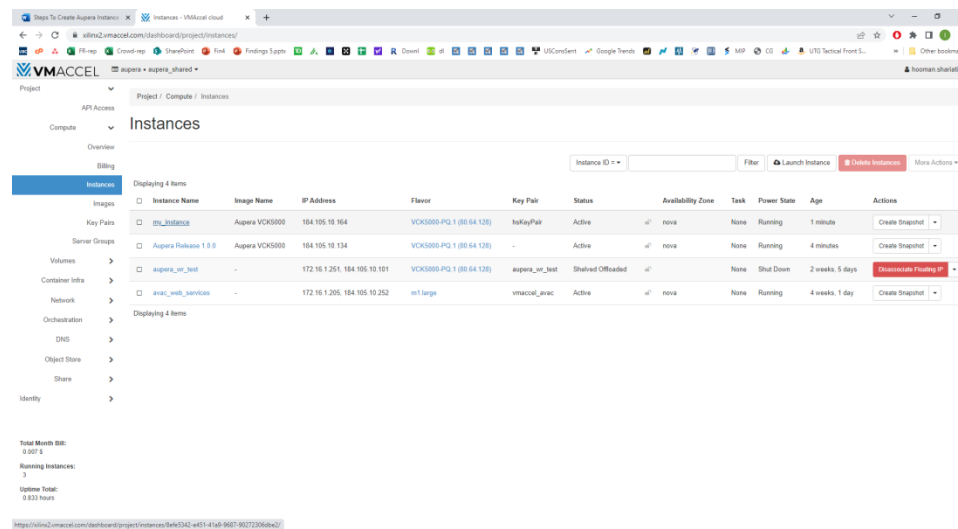


Figure 3.1. VMACCEL instances page – instance selection

B. Select the **Console** tab as Figure 3.2 shows.

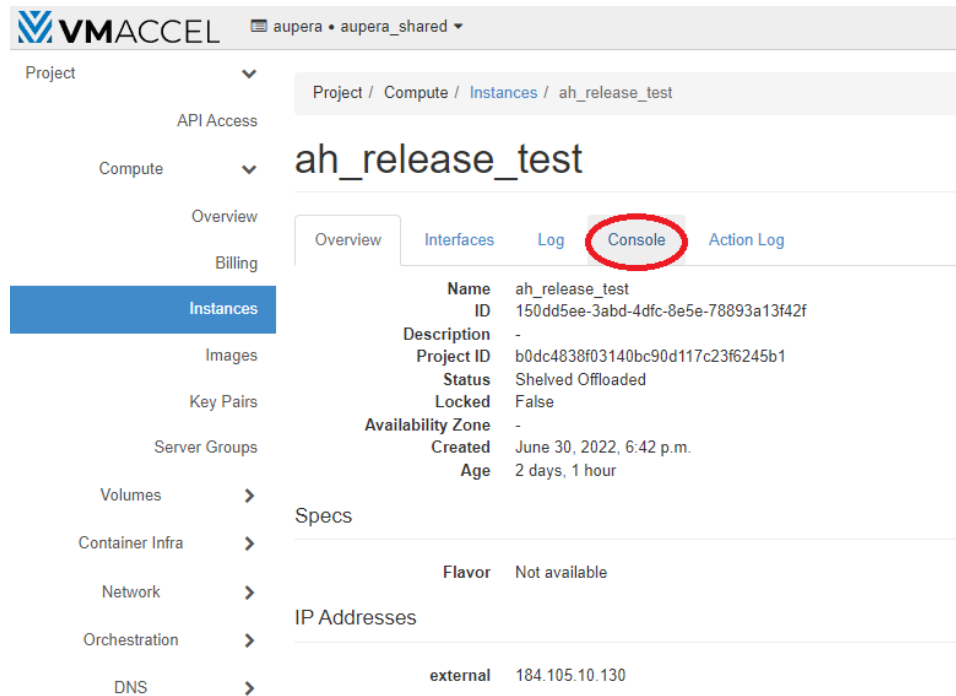


Figure 3.2. VMACcel instances page – instance console

C. Once the VNC window opens click on **Connect** as Figure 3.3 shows.

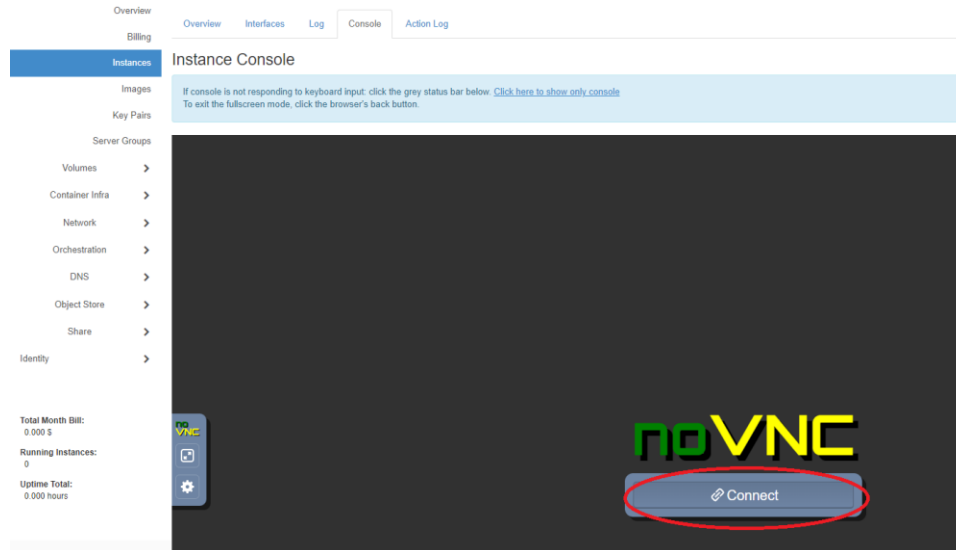


Figure 3.3. VMACcel console page – instance console connection

D. Inside the VMACcel VNC window, click on the Terminal icon to open a command line shell terminal.

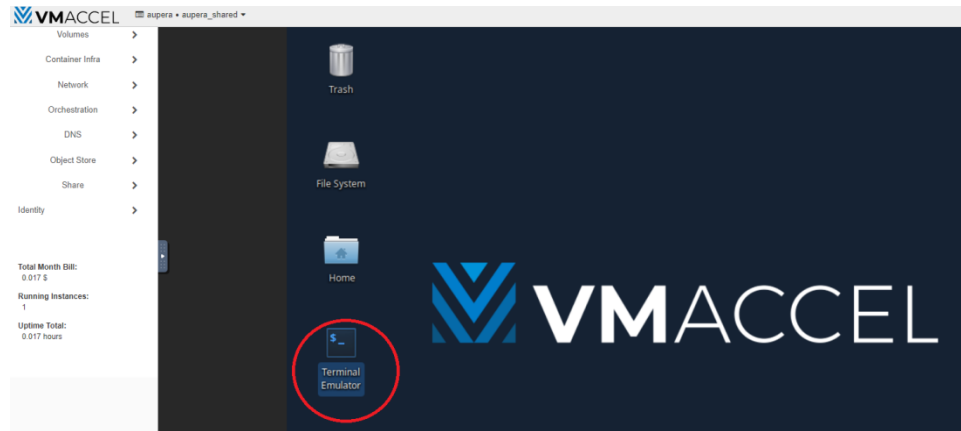


Figure 3.4. VMAccel console page – terminal emulator highlighted

E. From the terminal, you can enter the VMSS2.0 server's docker by running the command:

```
docker container exec -it aupera_server bash
```

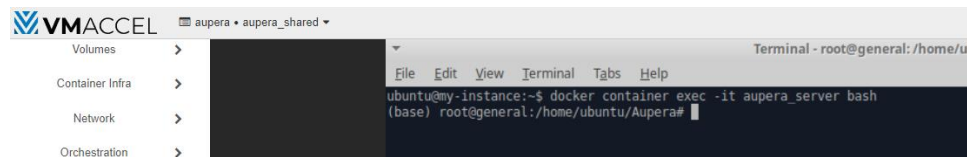


Figure 3.5. VMAccel console page – accessing VMSS2.0 server's docker

F. Once inside the docker, you will need to setup the environment (xbutil and vitis) by running the command below from any directory.

```
source set_env.sh
```

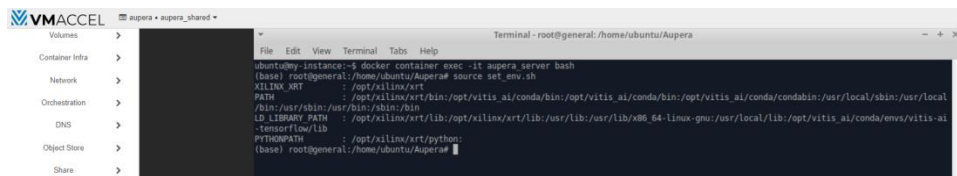


Figure 3.6. VMAccel console page – setting up software environment

At this point, the docker is ready to be used and you can proceed to the next section.

### 3.2 Running VMSS2.0 Pipelines

Generally, to run a VMSS2.0 pipeline, you can run the commands below from any directory inside of the VMSS 2.0 server docker. There are 3 ptxt files that are required to pass to *avaser*:

1. **Input:** comes after *-i* parameter and contains the same number of RTSP streams as the *input\_streams* contained in your *pipeline.ptxt*.

2. **Output:** comes after `-o` parameters and contains the same number of rtsp streams (or file passes) as the output\_streams contained in your pipeline.pbtxt.
3. **Config:** comes after `-c` parameter and contains your pipeline definition (the list of nodes and connections).

Below as an example of what the command should look like:

```
avaser -i input.pbtxt -o output.pbtxt -c pipeline.pbtxt
```

Below is an example of the content of an input.pbtxt file:

```
input_urls: "rtsp://10.10.190.114:554/key1"
input_urls: "rtsp://10.10.190.114:554/key2"
input_urls: "rtsp://10.10.190.114:554/key3"
```

Below is an example of the content of an output.pbtxt file:

```
output_urls: "rtsp://10.10.190.114:554/key4"
output_urls: "rtsp://10.10.190.114:554/key5"
output_urls: "/tmp/output_video_file.mp4"
```

**NOTE 1:** The output.pbtxt file could be empty if there are no output streams.

**NOTE 2:** The output.pbtxt file could contain file paths instead, in which case, the encoded video will be saved to disk instead of being sent to the RTSP streaming server.

### 3.3 Pipeline Examples

We have provided several examples of full pipelines [here](#). These are also included in the aupera\_server docker in the `/opt/aupera/avas/examples` folder. You can navigate to this location using the following command:

```
cd /opt/aupera/avas/examples
```

In most of the example folders there are two sets of pipelines pbtxt files: one called **using\_rtsp\_...pbtxt** and another called **using\_video.pbtxt**.

If you'd like to try the example pipelines on the sample videos, then all you need to do is to go inside the sub-folder of a specific example (box\_detector, box\_detector\_classifier\_cascade, or apl\_crowd\_flow) and run the following command:

```
avaser -i input.pbtxt -o output.pbtxt -c using_video.pbtxt
```

If you'd like to try the example pipelines on the RTSP streams that are automatically started by your VMAccel instance, then all you need to do is go inside the sub-folder of a specific example and run:

```
avaser -i input.pbtxt -o output.pbtxt -c using_rtsp.pbtxt
```



The results of our examples will be broad case in the IP address specified in the output.pbtxt file. In most cases, this is set to

```
output_urls: "rtsp://localhost:8554/out1"
```

which means that you can see the results by typing above RTSP URL into VLC; replacing “localhost” with the IP address of your VMAccel instance.

**NOTE 1:** If you’d like to run the pipelines on videos other than what we have provided, you will need to modify the “path” parameter in the video\_stream node. As shown in Figure 3.7 below:

```
node {
  name: "video_stream_node"
  calculator: "video_stream"
  input_stream: "inStream1"
  output_stream: "imgStream1080p"
  side_node_name: "crowd_flow"
  node_options: {
    [type.googleapis.com/gvis.VideoStreamOptions]: {
      path: "/opt/aupera/avas/examples/videos/C235-2021-04-21-13-13-04_first90S.mp4"
    }
  }
}
```

Figure 3.7. Aupera video stream output path in pbtxt file

**NOTE 2:** If you’d like to try our example pipelines on RTSP streams other than the ones lunched by your VMAccel instance, then you will need to edit the input.pbtxt files to set the input\_rtsp parameter to the URL of your RTSP streams.

For example, if the input.pbtxt of the example you are using contains:

```
input_urls: "rtsp://10.10.100.100:8554/stream1"
```

And the IP address of your VMAccel instance is 99.99.999.999:554/mystream, then you should edit your input.pbtxt to contain the following:

```
input_urls: "rtsp://99.99.999.999:554/mystream"
```

**NOTE 3:** To see the results of example pipeline on RTSP streams in other locations you will need to edit the output.pbtxt file in each folder to point either to the IP address of your RTSP sever; or to a valid file path. For example, if you’re inside the box\_detector example folder, and the IP address of the machine running your RTSP server is 10.10.100.100, then you will need to adjust the output.pbtxt to contain:

```
output_urls: "rtsp://10.10.100.100:8554/someKey"
```

In above case, you can watch the pipeline results at the RTSP stream provided above. Alternatively, your output.pbtxt could include line similar to the one as follows:

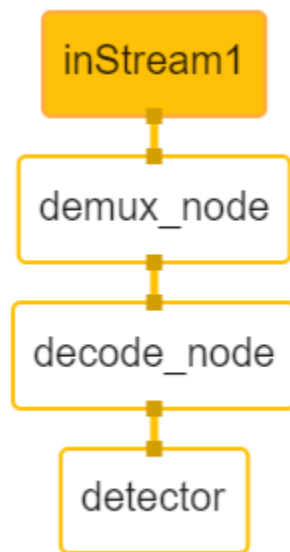
```
output_urls: "/tmp/video_output.mp4"
```

In above case, the pipeline's results will be saved to disk in a file accessible via the path specified above.

**NOTE 4:** Usually, you can kill the VMSS2.0 tasks (launched via the command line) by pressing CTL + C on your keyboard. For the throughput measurements tasks, you might need to press CTL + C a few times and / or use CTL + | combination to kill all the threads.

The pipeline examples that are included with the correct release are as follows:

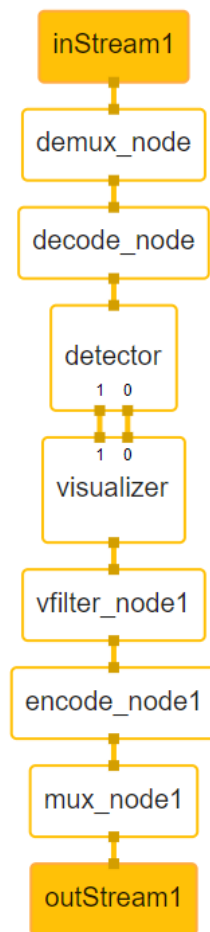
**A. box\_detector/using\_rtsp\_0output.pbt.txt**



**Figure 3.8. Aupera pipeline example with demux, decode, and detector nodes**

The pipeline in Figure 3.8 takes one input stream, runs a box\_detector network on the decoded frames, visualizes the detections on the frames, and saves the frames to disk (there is no output video).

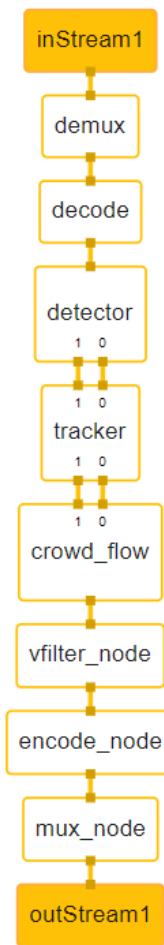
**B. box\_detector/using\_rtsp\_1output.pbt.txt**



**Figure 3.9. Aupera pipeline example with multiple nodes 1**

The pipeline in Figure 3.9 takes one input stream and one output stream. It runs a box\_detector network on the decoded frames and sends the detected bounding boxes and the frames to the box\_visualizer node. The box\_visualizer node, will visualize the detected bounding boxes on the frames and send them to video filter, video encoder, and mux nodes. The results are returned in an output rtsp stream or video file.

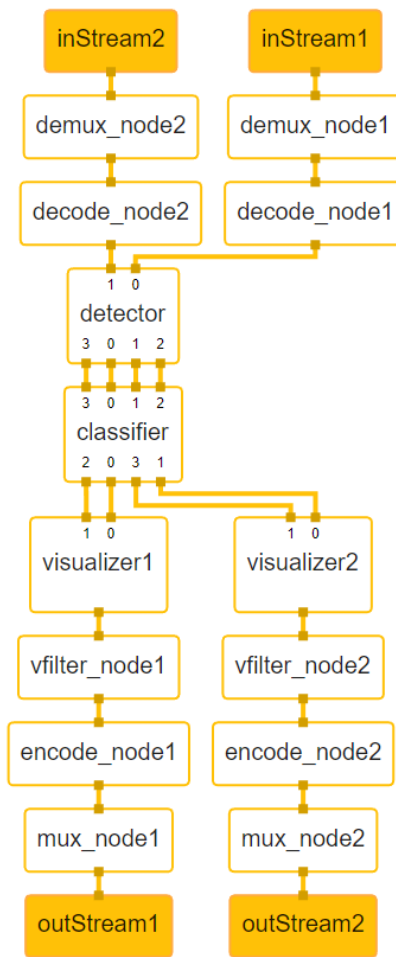
**C. apl\_crowd\_flow/using\_rtsp.pbt.txt :**



**Figure 3.10. Aupera pipeline example with multiple nodes 2**

The pipeline in Figure 3.10 takes one input stream and one output stream. It runs a box\_detector (at some interval), which passes the detected bounding boxes and the frames to a box\_Tracker. The box\_tracker tracks the objects (even on frames where the detector has not been run) and sends the bounding boxes and the frames to our crowd\_flow application node. This node applies the crowd\_flow logic; visualizes the results; and passes the frames to the video filter, encode, and mux nodes. The results can be seen in the output rtsp stream or a video file.

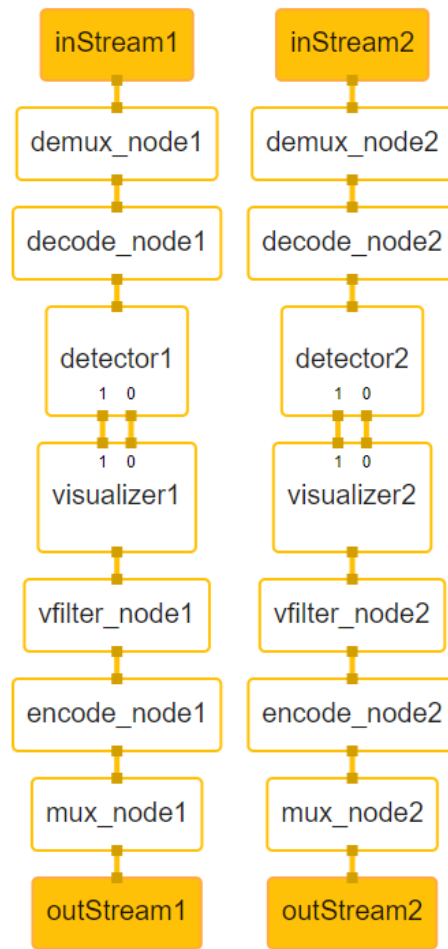
#### **D. box\_detector\_classifier\_cascade/using\_rtsp.pbt.txt**



**Figure 3.11. Aupera pipeline example with two input and output streams**

The pipeline in Figure 3.11 takes two synchronized input streams and produces two output streams. It runs a box detector (at some interval). Then it passes the frames in-tandem with the detections to the classifier node. The classifier node then classifies the objects detected by the detector node. It passes the classifications, which sends the results of each stream to its corresponding box\_visualizer node. The box\_visualizer will overlay the detections and classifications on the frames and send the results to video filter, stream encode, and stream mux nodes to be displayed over RTSP stream.

#### E. box\_detector/using\_rtsp.pbtxt



**Figure 3.12. Aupera pipeline example with two input and output streams**

The pipeline in Figure 3.12 is very similar to example B (figure 3.9); except it runs two detection tasks in parallel. This pipeline takes two input streams and produces two output streams. On stream 1, it runs a box detector with crowd models (head detection); while, on stream 2, it runs a box detector with retail models. The detection results are then passed to corresponding box\_visualizer nodes. The box\_visualizer nodes will overlay the detections on the frames and send the results to video filter, stream encode, and stream mux nodes to be displayed over RTSP stream.

## F. Throughput measurement using retail application

Similar to VMSS1.0, we use the retail application as an example pipeline for throughput measurement. This application consists of running a TinyYoloV3 object detector along with 3 resnet50 classification networks. All the networks are trained on the objects in the retail scenario. We also use the same retail.mp4 video as before (provided in the example\_videos folder). We maintain (and slightly exceed) the performance of

VMSS1.0 by supporting 37 streams (with I-frame-extraction) without any trackers and with 56 streams when using a tracker. We have, however, dramatically improved the accuracy of the pipeline when a tracker is used compared to VMSS1.0.

If you look inside the **example\_pipelines/throughput\_benchmarking/37streams**, you will find 3 configurations:

- **config\_noTracker\_noVideoOut.pbtxt** is the official test configuration. To run this test, you must ensure that the output.pbtxt file is empty.
- **config\_withTracker\_noVideoOut.pbtxt** performs the same test except the tracker is used. Here, we use a cluster size of 5 (i.e. classify each track 5 times) to achieve higher accuracy.
- **config\_withTracker\_withVideoOut.pbtxt** can be used to watch the visualized results on the output stream. To run this config, you will need to use the provided output.pbtxt with the correct output stream paths. Please note that since video visualization is a computationally expensive operation, it is only allowed while running the tracker. Also please note that when I-frame-extraction is true, the output video stream can only be saved at 10fps (since the video has an I-frame every 3<sup>rd</sup> frame). You can set I-frame\_extraction to false, in order to run and save the video at 30fps but that would require reducing the total number of streams (since we are making the test 3 times harder by passing 3 times the number of frames to the pipeline).

Finally, inside **example\_pipelines/throughput\_benchmarking/56streams** we have provided a single config (**config\_withTracker\_noVideoOut.pbtxt**), which can be used to confirm that the framework can run the retail pipeline with 56 streams without any frame drops. You can increase the **classifications\_cluster\_size** parameter to achieve higher classification accuracy if needed. Although, in this example, even with a value of 1, the classification accuracy is not far lower than running without a tracker. Essentially, this parameter specifies how many times we run the classification on each track. For example, when a value of 5 is specified, for each track, we run the classifier 5 times, and use the most frequent (i.e., the mode) classification as the final result.