`visionai`

VisionAI Toolkit

VisionAI tookit provides a large number of ready-to-deploy scenarios built using latest computer vision frameworks. Supports many of the common workplace health and safety use-cases.

Start by exploring scenarios through visionai scenario list command. After that, you can create a pipeline through the pipeline commands. Once a pipeline is configured, you can run the pipeline on the any number of cameras.

Running the toolkit does assume a NVIDIA GPU powered machine for efficient performance. Please see the system requirements on the documentation.

You can instead opt to install it through Azure Managed VM, with preconfigured machines & recommended hardware support. You can find information about this on our documentation website.

Visit https://docs.visionify.ai for more details.

Usage:

```
$ visionai [OPTIONS] COMMAND [ARGS]...
```

Options:

- `--verbose`: [default: False]
- `--version`: [default: False]
- `--install-completion`: Install completion for the current shell.
- `--show-completion`: Show completion for the current shell, to copy it or customize the installation.
- `--help`: Show this message and exit.

Commands:

```
╭─ Commands ──────────────────────────────────────────────────────╮
│ auth        Authentication commands                              │
│ camera      Add/remove/manage cameras                            │
│ device      Device commands                                      │
│ init        Initialize VisionAI library                          │
│ model       Manage models                                        │
│ pipeline    Manage pipelines                                     │
│ scenario    Add/remove scenarios to camera                       │
│ status       Print status of all running containers.          │
│ stop         Stop all running containers.                     │
│ web          Start/stop web server                            │
╰───────────────────────────────────────────────────────────────╯
```

`visionai auth`

Authorization (logging in/out)

Login and get authorization token etc.

You can login/logout check authorization token with this.

Usage:

```
$ visionai auth [OPTIONS] COMMAND [ARGS]...
```

Options:

- `--help`: Show this message and exit.

Commands:

- `login`: Login with an application token.
- `logout`: Logout from your session Get the auth token...
- `status`: Check login status Check the current login...

`visionai auth login`

Login with an application token.

Get the auth token from our website

Usage:

```
$ visionai auth login [OPTIONS]
```

Options:

- `--token TEXT`: Authenticate the app through token [required]
- `--help`: Show this message and exit.

`visionai auth logout`

Logout from your session

Get the auth token from our website

Usage:

```
$ visionai auth logout [OPTIONS]
```

Options:

- `--help`: Show this message and exit.

`visionai auth status`

Check login status

Check the current login system.

Usage:

```
$ visionai auth status [OPTIONS]
```

Options:

- `--help`: Show this message and exit.

`visionai camera`

Manage cameras

An organization can have multiple cameras that are installed at different places. They may be from different vendors and/or maybe using different security surveillance software. Most cameras however do support RTSP, RTMP or HLS streams as an output. Please refer to your camera vendor documentation to find this out. This module will help you onboard those cameras on visionai systems by using a simple named instance for each camera.

Usage:

```
$ visionai camera [OPTIONS] COMMAND [ARGS]...
```

Options:

- `--help`: Show this message and exit.

Commands:

- `add`: Add a named camera instance Add a camera as a...
- `add-scenario`: Add a scenario for a camera Add an individual...

- `list`: List available cameras Print cameras...
- `list-scenario`: List scenarios configured for a camera
- `list-scenarios`: List scenarios configured for a camera
- `preview`: Preview the camera system View the camera...
- `remove`: Remove a camera from the system Specify a...
- `remove-scenario`: Remove a scenario from a camera Specify a...
- `reset`: Reset all camera configuration.

`visionai camera add`

Add a named camera instance

Add a camera as a named instance in the system. For adding a camera we support RTSP, HLS, HTTP(S) systems. To add a camera you need to provide a name for the camera, URI for the camera (including any username/password within the URI itself), description for camera (about its location, where its pointing, who is the vendor etc.).

Before the camera is added - we need to test out if the camera instance is valid. We need to be able to read from the camera and calculate its FPS. Show this information on the screen.

Usage:

```
$ visionai camera add [OPTIONS]
```

Options:

- `--name TEXT`: Camera Name [required]
- `--uri TEXT`: URI for camera [required]
- `--description TEXT`: Description [required]
- `--help`: Show this message and exit.

`visionai camera add-scenario`

Add a scenario for a camera

Add an individual scenario to be run for a camera. Specify the names for scenario and camera.

Usage:

```
$ visionai camera add-scenario [OPTIONS]
```

Options:

- `--camera TEXT`: camera name [required]
- `--scenario TEXT`: scenario name [required]
- `--help`: Show this message and exit.

`visionai camera list`

List available cameras

Print cameras available in the system and the scenarios / routines that are set up for them.

Usage:

```
$ visionai camera list [OPTIONS]
```

Options:

- `--help`: Show this message and exit.

`visionai camera list-scenario`

List scenarios configured for a camera

Usage:

```
$ visionai camera list-scenario [OPTIONS]
```

Options:

- `--camera TEXT`: Camera name [default:]
- `--help`: Show this message and exit.

`visionai camera list-scenarios`

List scenarios configured for a camera

Usage:

```
$ visionai camera list-scenarios [OPTIONS]
```

Options:

- `--camera TEXT`: Camera name [default:]

- `--help`: Show this message and exit.

`visionai camera preview`

Preview the camera system

View the camera feed, review FPS etc available for camera.

Usage:

```
$ visionai camera preview [OPTIONS]
```

Options:

- `--name TEXT`: camera name to preview [required]
- `--help`: Show this message and exit.

`visionai camera remove`

Remove a camera from the system

Specify a named camera that needs to be removed from the system. Once removed, all the scenarios and pre-process routines associated with the camera will be removed.

Usage:

```
$ visionai camera remove [OPTIONS]
```

Options:

- `--name TEXT`: [default: Camera name]
- `--help`: Show this message and exit.

`visionai camera remove-scenario`

Remove a scenario from a camera

Specify a named scenario that needs to be removed from the system. Once removed, all the scenarios and pre-process routines associated with the scenario will be removed.

Usage:

```
$ visionai camera remove-scenario [OPTIONS]
```

Options:

- `--camera TEXT`: camera [required]
- `--scenario TEXT`: scenario name [required]
- `--help`: Show this message and exit.

`visionai camera reset`

Reset all camera configuration.

All cameras and their scenarios would be removed from the system. Any earlier configuration is backed up as a timed json backup file.

Usage:

```
$ visionai camera reset [OPTIONS]
```

Options:

- `--confirm / --no-confirm`: Confirm delete [default: False]
- `--help`: Show this message and exit.

`visionai cameras`

... alias for camera

Usage:

```
$ visionai cameras [OPTIONS] COMMAND [ARGS]...
```

Options:

- `--help`: Show this message and exit.

Commands:

- `add`: Add a named camera instance Add a camera as a...
- `add-scenario`: Add a scenario for a camera Add an individual...
- `list`: List available cameras Print cameras...
- `list-scenario`: List scenarios configured for a camera
- `list-scenarios`: List scenarios configured for a camera
- `preview`: Preview the camera system View the camera...
- `remove`: Remove a camera from the system Specify a...

- `remove-scenario`: Remove a scenario from a camera Specify a...
- `reset`: Reset all camera configuration.

`visionai cameras add`

Add a named camera instance

Add a camera as a named instance in the system. For adding a camera we support RTSP, HLS, HTTP(S) systems. To add a camera you need to provide a name for the camera, URI for the camera (including any username/password within the URI itself), description for camera (about its location, where its pointing, who is the vendor etc.).

Before the camera is added - we need to test out if the camera instance is valid. We need to be able to read from the camera and calculate its FPS. Show this information on the screen.

Usage:

```
$ visionai cameras add [OPTIONS]
```

Options:

- `--name TEXT`: Camera Name [required]
- `--uri TEXT`: URI for camera [required]
- `--description TEXT`: Description [required]
- `--help`: Show this message and exit.

`visionai cameras add-scenario`

Add a scenario for a camera

Add an individual scenario to be run for a camera. Specify the names for scenario and camera.

Usage:

```
$ visionai cameras add-scenario [OPTIONS]
```

Options:

- `--camera TEXT`: camera name [required]
- `--scenario TEXT`: scenario name [required]
- `--help`: Show this message and exit.

`visionai cameras list`

List available cameras

Print cameras available in the system and the scenarios / routines that are set up for them.

Usage:

```
$ visionai cameras list [OPTIONS]
```

Options:

- `--help`: Show this message and exit.

`visionai cameras list-scenario`

List scenarios configured for a camera

Usage:

```
$ visionai cameras list-scenario [OPTIONS]
```

Options:

- `--camera TEXT`: Camera name [default:]
- `--help`: Show this message and exit.

`visionai cameras list-scenarios`

List scenarios configured for a camera

Usage:

```
$ visionai cameras list-scenarios [OPTIONS]
```

Options:

- `--camera TEXT`: Camera name [default:]
- `--help`: Show this message and exit.

`visionai cameras preview`

Preview the camera system

View the camera feed, review FPS etc available for camera.

Usage:

```
$ visionai cameras preview [OPTIONS]
```

Options:

- `--name TEXT`: camera name to preview [required]
- `--help`: Show this message and exit.

`visionai cameras remove`

Remove a camera from the system

Specify a named camera that needs to be removed from the system. Once removed, all the scenarios and pre-process routines associated with the camera will be removed.

Usage:

```
$ visionai cameras remove [OPTIONS]
```

Options:

- `--name TEXT`: [default: Camera name]
- `--help`: Show this message and exit.

`visionai cameras remove-scenario`

Remove a scenario from a camera

Specify a named scenario that needs to be removed from the system. Once removed, all the scenarios and pre-process routines associated with the scenario will be removed.

Usage:

```
$ visionai cameras remove-scenario [OPTIONS]
```

Options:

- `--camera TEXT`: camera [required]
- `--scenario TEXT`: scenario name [required]
- `--help`: Show this message and exit.

`visionai cameras reset`

Reset all camera configuration.

All cameras and their scenarios would be removed from the system. Any earlier configuration is backed up as a timed json backup file.

Usage:

```
$ visionai cameras reset [OPTIONS]
```

Options:

- `--confirm / --no-confirm`: Confirm delete [default: False]
- `--help`: Show this message and exit.

`visionai device`

Manage device features

Since scenarios run on individual edge-devices, and we don't have enough control over the CPU, Memory, GPU statistics - it is imperative that we have strong methods for validating if a scenario can run on a chosen platform. This module provides many utilities to check CPU, Memory and GPU statistics for the edge device. We also provide an Azure Managed service where these scenarios can be configured and run on your premise on pre-validated VM machines.

Usage:

```
$ visionai device [OPTIONS] COMMAND [ARGS]...
```

Options:

- `--help`: Show this message and exit.

Commands:

- `list`: List available devices Get a list of all...
- `modules`: List running modules on the device Again this...
- `select`: Select a device Not sure why is this needed...
- `stats`: Machine health (GPU/Mem stats) Show machine...

`visionai device list`

List available devices

Get a list of all available [processing] devices

Usage:

```
$ visionai device list [OPTIONS]
```

Options:

- `--help`: Show this message and exit.

`visionai device modules`

List running modules on the device

Again this does not make much sense at this time. Let's revisit.

Usage:

```
$ visionai device modules [OPTIONS]
```

Options:

- `--help`: Show this message and exit.

`visionai device select`

Select a device

Not sure why is this needed at this time.

Usage:

```
$ visionai device select [OPTIONS] DEVICE
```

Arguments:

- `DEVICE`: [required]

Options:

- `--help`: Show this message and exit.

`visionai device stats`

Machine health (GPU/Mem stats)

Show machine health (GPU/memory stats). This can be used to determine if more scenarios can be run on the machine or not.

Usage:

```
$ visionai device stats [OPTIONS]
```

Options:

- `--help`: Show this message and exit.

`visionai devices`

... alias for device

Usage:

```
$ visionai devices [OPTIONS] COMMAND [ARGS]...
```

Options:

- `--help`: Show this message and exit.

Commands:

- `list`: List available devices Get a list of all...
- `modules`: List running modules on the device Again this...
- `select`: Select a device Not sure why is this needed...
- `stats`: Machine health (GPU/Mem stats) Show machine...

`visionai devices list`

List available devices

Get a list of all available [processing] devices

Usage:

```
$ visionai devices list [OPTIONS]
```

Options:

- `--help`: Show this message and exit.

`visionai devices modules`

List running modules on the device

Again this does not make much sense at this time. Let's revisit.

Usage:

```
$ visionai devices modules [OPTIONS]
```

Options:

- `--help`: Show this message and exit.

`visionai devices select`

Select a device

Not sure why is this needed at this time.

Usage:

```
$ visionai devices select [OPTIONS] DEVICE
```

Arguments:

- `DEVICE`: [required]

Options:

- `--help`: Show this message and exit.

`visionai devices stats`

Machine health (GPU/Mem stats)

Show machine health (GPU/memory stats). This can be used to determine if more scenarios can be run on the machine or not.

Usage:

```
$ visionai devices stats [OPTIONS]
```

Options:

- `--help`: Show this message and exit.

`visionai model`

Serve models

Before we can run any scenarios - the models necessary for them must be ready. We use Triton inference server to make the best use of GPU/CPU resources available on the machine in order to serve our models. Any models that are available in models-repo folder would be served after this (TODO - only serve models configured in scenarios).

Usage:

```
$ visionai model [OPTIONS] COMMAND [ARGS]...
```

Options:

- `--help`: Show this message and exit.

Commands:

- `check`: Check model-server status & print helpful...
- `serve`: Start serving all available models.
- `start`: Start serving all available models.
- `status`: Show the status of serving models Shows how...
- `stop`: Stop serving all models.

`visionai model check`

Check model-server status & print helpful debug info.

TODO: Goal of the check command is to identify any configuration/dependency issues that we can inform to user that he can fix on his end. This could be like missing dependency, missing software package, missing driver details etc.

- Check if model-server is running or not.
- Check if triton-client can access model-server
- Check what are the models served
- Print all of this in a pretty manner [checkbox based]
- Check container logs & show them here.
- grep container logs for common errors & highlight that in output

Usage:

```
$ visionai model check [OPTIONS]
```

Options:

- `--help`: Show this message and exit.

`visionai model serve`

Start serving all available models.

All models present in the models-repo/ will be served. We use triton inference server to serve them. The triton server will be at http://localhost:8000, grpc://localhost:8001.

Please make sure these two ports are not used by anyone else.

Usage:

```
$ visionai model serve [OPTIONS]
```

Options:

- `--help`: Show this message and exit.

`visionai model start`

Start serving all available models.

All models present in the models-repo/ will be served. We use triton inference server to serve them. The triton server will be at http://localhost:8000, grpc://localhost:8001.

Please make sure these two ports are not used by anyone else.

Usage:

```
$ visionai model start [OPTIONS]
```

Options:

- `--help`: Show this message and exit.

`visionai model status`

Show the status of serving models

Shows how many models are being served, metrics for the models etc.

Usage:

```
$ visionai model status [OPTIONS]
```

Options:

- `--help`: Show this message and exit.

`visionai model stop`

Stop serving all models.

This method will stop serving all models. Any inference running will all be stopped as well.

Usage:

```
$ visionai model stop [OPTIONS]
```

Options:

- `--help`: Show this message and exit.

`visionai models`

... alias for model

Usage:

```
$ visionai models [OPTIONS] COMMAND [ARGS]...
```

Options:

- `--help`: Show this message and exit.

Commands:

- `check`: Check model-server status & print helpful...
- `serve`: Start serving all available models.
- `start`: Start serving all available models.
- `status`: Show the status of serving models Shows how...
- `stop`: Stop serving all models.

`visionai models check`

Check model-server status & print helpful debug info.

TODO: Goal of the check command is to identify any configuration/dependency issues that we can inform to user that he can fix on his end. This could be like missing dependency, missing software package, missing driver details etc.

- Check if model-server is running or not.
- Check if triton-client can access model-server
- Check what are the models served

- Print all of this in a pretty manner [checkbox based]
- Check container logs & show them here.
- grep container logs for common errors & highlight that in output

Usage:

```
$ visionai models check [OPTIONS]
```

Options:

- `--help`: Show this message and exit.

`visionai models serve`

Start serving all available models.

All models present in the models-repo/ will be served. We use triton inference server to serve them. The triton server will be at http://localhost:8000, grpc://localhost:8001.

Please make sure these two ports are not used by anyone else.

Usage:

```
$ visionai models serve [OPTIONS]
```

Options:

- `--help`: Show this message and exit.

`visionai models start`

Start serving all available models.

All models present in the models-repo/ will be served. We use triton inference server to serve them. The triton server will be at http://localhost:8000, grpc://localhost:8001.

Please make sure these two ports are not used by anyone else.

Usage:

```
$ visionai models start [OPTIONS]
```

Options:

- `--help`: Show this message and exit.

`visionai models status`

Show the status of serving models

Shows how many models are being served, metrics for the models etc.

Usage:

```
$ visionai models status [OPTIONS]
```

Options:

- `--help`: Show this message and exit.

`visionai models stop`

Stop serving all models.

This method will stop serving all models. Any inference running will all be stopped as well.

Usage:

```
$ visionai models stop [OPTIONS]
```

Options:

- `--help`: Show this message and exit.

`visionai pipeline`

Manage pipelines

Pipeline is a sequence of preprocess routines and scenarios to be run on a given set of cameras. Each pipeline can be configured to run specific scenarios - each scenario with their own customizations for event notifications. This module provides robust methods for managing pipelines, showing their details, adding/remove cameras from pipelines and running a pipeline.

Usage:

```
$ visionai pipeline [OPTIONS] COMMAND [ARGS]...
```

Options:

- `--help`: Show this message and exit.

Commands:

- `add-camera`: Add a camera to a pipeline Each pipeline...
- `add-preprocess`: Add a preprocess routine to a pipeline...
- `add-scenario`: Add a scenario to a pipeline The order of the...
- `create`: Create a named pipeline Create a named...
- `remove-camera`: Remove a camera from a pipeline This method...
- `reset`: Reset the pipeline to original state.
- `run`: Run a pipeline of scenarios on given cameras...
- `show`: Show details of a pipeline Show what is...

`visionai pipeline add-camera`

Add a camera to a pipeline

Each pipeline consists of a bunch of scenarios to run and which cameras they need to be run on. This method allows the user to add one or more named camera instance to a pipeline. Please note the camera instance has to be created prior to adding it here.

add a camera

$ visionai camera add --name OFFICE-01 --uri https://youtube.com

add camera to pipeline

$ visionai pipeline --name test_pipe add-camera --name OFFICE-01

@arg pipeline - specify a named pipeline @arg camera - specify name of the camera to add

@return None

Usage:

```
$ visionai pipeline add-camera [OPTIONS]
```

Options:

- `--pipeline TEXT`: pipeline name [required]
- `--camera TEXT`: camera to add [required]

- `--help`: Show this message and exit.

`visionai pipeline add-preprocess`

Add a preprocess routine to a pipeline

Preprocessing tasks are run prior to scenarios. The order in which multiple preprocess tasks are added does not matter. All added preprocess routines are executed in different threads.

$ visionai pipeline --name test_pipe add-preprocess --name face-blur

$ visionai pipeline --name test_pipe add-preprocess --name text-blur

$ visionai pipeline --name test_pipe show

$ visionai pipeline --name test_pipe run

@arg pipeline - specify a named pipeline @arg preprocess - specify name of the preprocess task to run

@return None

Usage:

```
$ visionai pipeline add-preprocess [OPTIONS]
```

Options:

- `--pipeline TEXT`: pipeline name [required]
- `--preprocess TEXT`: preprocess routine to add [required]
- `--help`: Show this message and exit.

`visionai pipeline add-scenario`

Add a scenario to a pipeline

The order of the scenarios does not matter. All added scenarios are run in different threads. All scenarios are run after pre-processing stage is done.

$visionai pipeline --name test_pipe add-scenario --name smoke-and-fire

$visionai pipeline --name test_pipe add-scenario --name ppe-detection

$visionai pipeline --name test_pipe run

@arg pipeline - specify a named pipeline @arg scenario - specify name of the scenario to run

@return None

Usage:

```
$ visionai pipeline add-scenario [OPTIONS]
```

Options:

- `--pipeline TEXT`: pipeline name [required]
- `--scenario TEXT`: scenario to add [required]
- `--help`: Show this message and exit.

`visionai pipeline create`

Create a named pipeline

Create a named pipeline. Pipeline is a list of scenarios to be run for specific cameras. The flow is as follows. Create a pipeline using:

visionai pipeline create --name test_pipe

visionai pipeline add-scenario --pipeline test_pipe --name smoke-and-fire

visionai pipeline add-scenario --pipeline test_pipe --name ppe-detection

visionai pipeline add-preprocess --pipeline test_pipe --name face-blur

visionai pipeline add-preprocess --pipeline test_pipe --name text-blur

visionai pipeline add-scenario --pipeline test_pipe --name max-occupancy

visionai pipeline show --pipeline test_pipe

visionai pipeline add-camera --pipeline test_pipe --name CAMERA-01

visionai pipeline add-camera --pipeline test_pipe --name CAMERA-02

visionai pipeline show --pipeline test_pipe

visionai pipeline run --pipeline test_pipe

@arg pipeline - specify a named pipeline

@return None

Usage:

```
$ visionai pipeline create [OPTIONS]
```

Options:

- `--name TEXT`: pipeline name [required]
- `--help`: Show this message and exit.

`visionai pipeline remove-camera`

Remove a camera from a pipeline

This method can be used to remove a camera from a pipeline.

$ visionai pipeline --name test_pipe remove-camera --name OFFICE-01

@arg pipeline - specify a named pipeline @arg camera - specify name of the camera to remove

@return None

Usage:

```
$ visionai pipeline remove-camera [OPTIONS]
```

Options:

- `--pipeline TEXT`: pipeline name [required]
- `--camera TEXT`: camera to remove [required]
- `--help`: Show this message and exit.

`visionai pipeline reset`

Reset the pipeline to original state.

Deletes all cameras, scenarios and scenario configuration from the pipeline. Its as if the pipeline has been deleted and created from scratch again.

$ visionai pipeline --name test_pipe reset

@arg pipeline - pipeline to reset

@return None

Usage:

```
$ visionai pipeline reset [OPTIONS]
```

Options:

- `--pipeline TEXT`: pipeline name [required]

* `--help`: Show this message and exit.

`visionai pipeline run`

Run a pipeline of scenarios on given cameras

Specify different scenarios to run on one or more cameras. This method can be directly used to specify scenarios and cameras directly. Else you can configure a named pipeline and then run it here.

@arg pipeline - specify a named pipeline

@return None

Usage:

```
$ visionai pipeline run [OPTIONS]
```

Options:

* `--pipeline TEXT`: Pipeline to run [required]
* `--help`: Show this message and exit.

`visionai pipeline show`

Show details of a pipeline

Show what is configured in the current pipeline.

$ visionai pipeline --name test_pipe show

@arg pipeline - specify a named pipeline

@return None

Usage:

```
$ visionai pipeline show [OPTIONS]
```

Options:

* `--pipeline TEXT`: pipeline name [required]
* `--help`: Show this message and exit.

`visionai pipelines`

... alias for pipeline

Usage:

```
$ visionai pipelines [OPTIONS] COMMAND [ARGS]...
```

Options:

- `--help`: Show this message and exit.

Commands:

- `add-camera`: Add a camera to a pipeline Each pipeline...
- `add-preprocess`: Add a preprocess routine to a pipeline...
- `add-scenario`: Add a scenario to a pipeline The order of the...
- `create`: Create a named pipeline Create a named...
- `remove-camera`: Remove a camera from a pipeline This method...
- `reset`: Reset the pipeline to original state.
- `run`: Run a pipeline of scenarios on given cameras...
- `show`: Show details of a pipeline Show what is...

`visionai pipelines add-camera`

Add a camera to a pipeline

Each pipeline consists of a bunch of scenarios to run and which cameras they need to be run on. This method allows the user to add one or more named camera instance to a pipeline. Please note the camera instance has to be created prior to adding it here.

add a camera

$ visionai camera add --name OFFICE-01 --uri https://youtube.com

add camera to pipeline

$ visionai pipeline --name test_pipe add-camera --name OFFICE-01

@arg pipeline - specify a named pipeline @arg camera - specify name of the camera to add

@return None

Usage:

```
$ visionai pipelines add-camera [OPTIONS]
```

Options:

- `--pipeline TEXT`: pipeline name [required]
- `--camera TEXT`: camera to add [required]
- `--help`: Show this message and exit.

`visionai pipelines add-preprocess`

Add a preprocess routine to a pipeline

Preprocessing tasks are run prior to scenarios. The order in which multiple preprocess tasks are added does not matter. All added preprocess routines are executed in different threads.

$ visionai pipeline --name test_pipe add-preprocess --name face-blur

$ visionai pipeline --name test_pipe add-preprocess --name text-blur

$ visionai pipeline --name test_pipe show

$ visionai pipeline --name test_pipe run

@arg pipeline - specify a named pipeline @arg preprocess - specify name of the preprocess task to run

@return None

Usage:

```
$ visionai pipelines add-preprocess [OPTIONS]
```

Options:

- `--pipeline TEXT`: pipeline name [required]
- `--preprocess TEXT`: preprocess routine to add [required]
- `--help`: Show this message and exit.

`visionai pipelines add-scenario`

Add a scenario to a pipeline

The order of the scenarios does not matter. All added scenarios are run in different threads. All scenarios are run after pre-processing stage is done.

$visionai pipeline --name test_pipe add-scenario --name smoke-and-fire

$visionai pipeline --name test_pipe add-scenario --name ppe-detection

$visionai pipeline --name test_pipe run

@arg pipeline - specify a named pipeline @arg scenario - specify name of the scenario to run

@return None

Usage:

```
$ visionai pipelines add-scenario [OPTIONS]
```

Options:

- `--pipeline TEXT`: pipeline name [required]
- `--scenario TEXT`: scenario to add [required]
- `--help`: Show this message and exit.

`visionai pipelines create`

Create a named pipeline

Create a named pipeline. Pipeline is a list of scenarios to be run for specific cameras. The flow is as follows. Create a pipeline using:

visionai pipeline create --name test_pipe

visionai pipeline add-scenario --pipeline test_pipe --name smoke-and-fire

visionai pipeline add-scenario --pipeline test_pipe --name ppe-detection

visionai pipeline add-preprocess --pipeline test_pipe --name face-blur

visionai pipeline add-preprocess --pipeline test_pipe --name text-blur

visionai pipeline add-scenario --pipeline test_pipe --name max-occupancy

visionai pipeline show --pipeline test_pipe

visionai pipeline add-camera --pipeline test_pipe --name CAMERA-01

visionai pipeline add-camera --pipeline test_pipe --name CAMERA-02

visionai pipeline show --pipeline test_pipe visionai pipeline run --pipeline test_pipe

@arg pipeline - specify a named pipeline

@return None

Usage:

```
$ visionai pipelines create [OPTIONS]
```

Options:

- `--name TEXT`: pipeline name [required]
- `--help`: Show this message and exit.

`visionai pipelines remove-camera`

Remove a camera from a pipeline

This method can be used to remove a camera from a pipeline.

$ visionai pipeline --name test_pipe remove-camera --name OFFICE-01

@arg pipeline - specify a named pipeline @arg camera - specify name of the camera to remove

@return None

Usage:

```
$ visionai pipelines remove-camera [OPTIONS]
```

Options:

- `--pipeline TEXT`: pipeline name [required]
- `--camera TEXT`: camera to remove [required]
- `--help`: Show this message and exit.

`visionai pipelines reset`

Reset the pipeline to original state.

Deletes all cameras, scenarios and scenario configuration from the pipeline. Its as if the pipeline has been deleted and created from scratch again.

$ visionai pipeline --name test_pipe reset

@arg pipeline - pipeline to reset

@return None

Usage:

```
$ visionai pipelines reset [OPTIONS]
```

Options:

- `--pipeline TEXT`: pipeline name [required]
- `--help`: Show this message and exit.

`visionai pipelines run`

Run a pipeline of scenarios on given cameras

Specify different scenarios to run on one or more cameras. This method can be directly used to specify scenarios and cameras directly. Else you can configure a named pipeline and then run it here.

@arg pipeline - specify a named pipeline

@return None

Usage:

```
$ visionai pipelines run [OPTIONS]
```

Options:

- `--pipeline TEXT`: Pipeline to run [required]
- `--help`: Show this message and exit.

`visionai pipelines show`

Show details of a pipeline

Show what is configured in the current pipeline.

$ visionai pipeline --name test_pipe show

@arg pipeline - specify a named pipeline

@return None

Usage:

```
$ visionai pipelines show [OPTIONS]
```

Options:

- `--pipeline TEXT`: pipeline name [required]
- `--help`: Show this message and exit.

`visionai scenario`

Manage scenarios

An organization can have multiple scenarios that are installed at different places. They may be from different vendors and/or maybe using different security surveillance software. Most scenarios however do support RTSP, RTMP or HLS streams as an output. Please refer to your scenario vendor documentation to find this out. This module will help you onboard those scenarios on visionai systems by using a simple named instance for each scenario.

Usage:

```
$ visionai scenario [OPTIONS] COMMAND [ARGS]...
```

Options:

- `--help`: Show this message and exit.

Commands:

- `download`: Download models for scenarios Ex: visionai...
- `list`: List all scenarios available List all...
- `test`: Run the scenario locally to test it out.

`visionai scenario download`

Download models for scenarios

Ex: visionai scenario download ppe-detection # download ppe-detection scenario Ex: visionai scenario download all # download all configured scenarios for the org Ex: visionai scenario download world # download all available scenarios

Download models for a given scenario, or download models for all scenarios that have been configured.

Usage:

```
$ visionai scenario download [OPTIONS] NAME
```

Arguments:

* `NAME`: [required]

Options:

* `--help`: Show this message and exit.

`visionai scenario list`

List all scenarios available

List all scenarios available in the system. This includes scenarios that may or maynot be applied to any specific camera.

Usage:

```
$ visionai scenario list [OPTIONS]
```

Options:

* `--help`: Show this message and exit.

`visionai scenario test`

Run the scenario locally to test it out.

* Download the model if not available.
* Pull the model server container image.
* Start the model server container with this model.
* Run inference with this model

Usage:

```
$ visionai scenario test [OPTIONS] NAME
```

Arguments:

* `NAME`: [required]

Options:

* `--camera TEXT`: Camera name (default is webcam) [default: 0]
* `--help`: Show this message and exit.

`visionai scenarios`

... alias for scenario

Usage:

```
$ visionai scenarios [OPTIONS] COMMAND [ARGS]...
```

Options:

- `--help`: Show this message and exit.

Commands:

- `download`: Download models for scenarios Ex: visionai...
- `list`: List all scenarios available List all...
- `test`: Run the scenario locally to test it out.

`visionai scenarios download`

Download models for scenarios

Ex: visionai scenario download ppe-detection # download ppe-detection scenario Ex: visionai scenario download all # download all configured scenarios for the org Ex: visionai scenario download world # download all available scenarios

Download models for a given scenario, or download models for all scenarios that have been configured.

Usage:

```
$ visionai scenarios download [OPTIONS] NAME
```

Arguments:

- `NAME`: [required]

Options:

- `--help`: Show this message and exit.

`visionai scenarios list`

List all scenarios available

List all scenarios available in the system. This includes scenarios that may or maynot be applied to any specific camera.

Usage:

```
$ visionai scenarios list [OPTIONS]
```

Options:

- `--help`: Show this message and exit.

`visionai scenarios test`

Run the scenario locally to test it out.

- Download the model if not available.
- Pull the model server container image.
- Start the model server container with this model.
- Run inference with this model

Usage:

```
$ visionai scenarios test [OPTIONS] NAME
```

Arguments:

- `NAME`: [required]

Options:

- `--camera TEXT`: Camera name (default is webcam) [default: 0]
- `--help`: Show this message and exit.

`visionai ui`

Start/stop web-app

Start and stop the VisionAI web-app which can be a more intuitive way of managing cameras, pipelines and scenarios. Web-app also provides a live-stream view of the cameras.

Usage:

```
$ visionai ui [OPTIONS] COMMAND [ARGS]...
```

Options:

- `--help`: Show this message and exit.

Commands:

- `start`: Start web server Use this function to start...
- `status`: Web service status Use this function to get...
- `stop`: Stop web server Use this function to stop...

`visionai ui start`

Start web server

Use this function to start the web-service. Web service can be used for more intuitive configuration for the cameras and scenarios. Web-app is also the place to view event details, camera live-stream etc.

Usage:

```
$ visionai ui start [OPTIONS]
```

Options:

- `--help`: Show this message and exit.

`visionai ui status`

Web service status

Use this function to get the status of the web-service. (if its running or not. This function also prints diagnostic information like last few log messages etc.)

Usage:

```
$ visionai ui status [OPTIONS]
```

Options:

- `--tail INTEGER`: tail number of lines [default: 20]
- `--help`: Show this message and exit.

`visionai ui stop`

Stop web server

Use this function to stop already running web-service. There can be a single instance of the web-service supported currently. So there is no need for any arguments for this function.

Usage:

```
$ visionai ui stop [OPTIONS]
```

Options:

- `--web TEXT`
- `--help`: Show this message and exit.

`visionai web`

... alias for ui

Usage:

```
$ visionai web [OPTIONS] COMMAND [ARGS]...
```

Options:

- `--help`: Show this message and exit.

Commands:

- `start`: Start web server Use this function to start...
- `status`: Web service status Use this function to get...
- `stop`: Stop web server Use this function to stop...

`visionai web start`

Start web server

Use this function to start the web-service. Web service can be used for more intuitive configuration for the cameras and scenarios. Web-app is also the place to view event details, camera live-stream etc.

Usage:

```
$ visionai web start [OPTIONS]
```

Options:

- `--help`: Show this message and exit.

`visionai web status`

Web service status

Use this function to get the status of the web-service. (if its running or not. This function also prints diagnostic information like last few log messages etc.)

Usage:

```
$ visionai web status [OPTIONS]
```

Options:

- `--tail INTEGER`: tail number of lines [default: 20]
- `--help`: Show this message and exit.

`visionai web stop`

Stop web server

Use this function to stop already running web-service. There can be a single instance of the web-service supported currently. So there is no need for any arguments for this function.

Usage:

```
$ visionai web stop [OPTIONS]
```

Options:

- `--web TEXT`
- `--help`: Show this message and exit.