

# Trame - the basics

Sebastien Jourdain

# Outline (~2 hours)

## ◆ Introduction

- What is trame?
- How trame works?
- Why is it revolutionary?

## ◆ Getting started

- Documentation
- Tutorial & examples
- Cookie cutter

## ◆ Trame and vtk(.js)

- Usage exploration

## ◆ Cheat sheets

- Syntax explained

## ◆ Cookiecutter

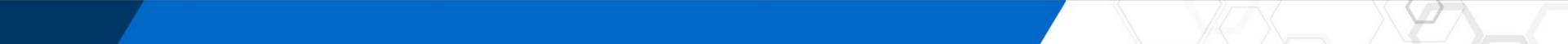
- Standardisation for python

## ◆ ParaView gotcha

- pypython is not your python

## ◆ Deployment

- Desktop, jupyter and cloud

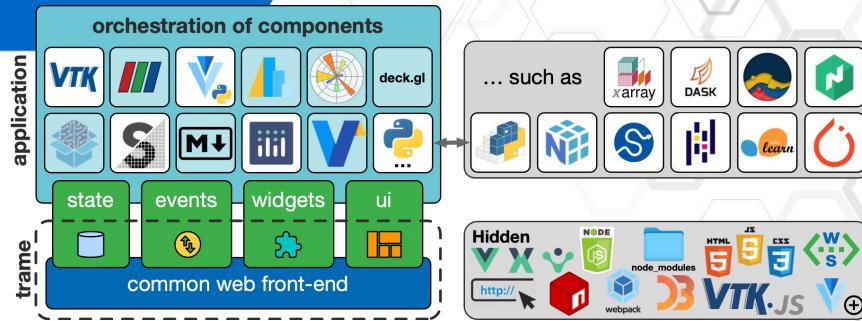


# Introduction



15 minutes

# What is trame?



- **Simple**

All the logic and UI definition can be done in plain Python

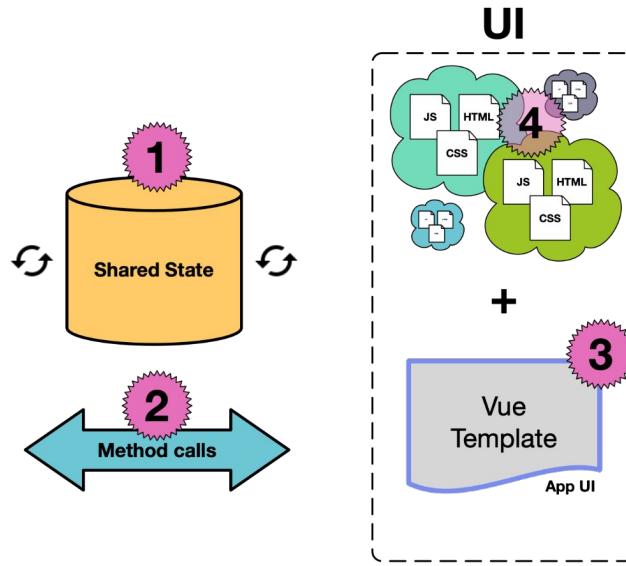
- **Powerful**

Python offers scientific and information data visualization with capable data processing (numpy, Plotly, Matplotlib, VTK, ParaView...)

- **Ubiquitous**

Runs on laptops, desktops, clusters, and the cloud while displaying everywhere (phone, tablet, laptop, workstation)

# How trame works?



- 0 - Just a Python file
- 1 - Simple data exchange
- 2 - Simple code binding
- 3 - Efficient UI definition
- 4 - Add-on widgets



**Client side**

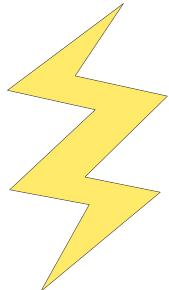
# Why is it revolutionary?

## New way



Just a Python application

No need to fix build system every 6 months.  
Only your python code to maintain and deploy.  
Uniform and simple way for coding and binding UI

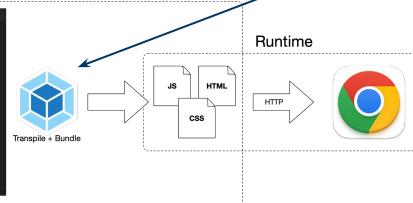


## Old way

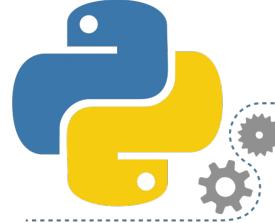
### App & Component Development

```
ApComment.vue
User: ~$ cd my-domain > Desktop > ApComment.vue > ...
1 <template>
2   <div>
3     <input type="text" v-model="aps" />
4     <div v-show="age < 21">
5       You should not drink especially in the US...
6     </div>
7     <div v-show="age >= 21">
8       Let's go after!
9     </div>
10    </div>
11  </template>
12
13  <script>
14    export default {
15      data() {
16        return {
17          age: 18,
18        }
19      }
20    </script>
```

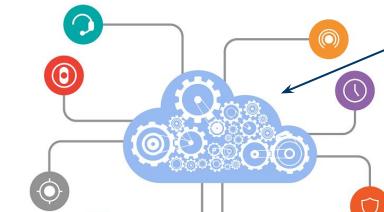
Transpile + Bundle



Break every 6 months



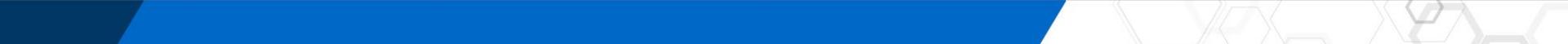
Complex client/server state  
management and binding



Maintenance cost

Complex process





# Getting started



5 minutes

# Getting started

## ◆ Documentation

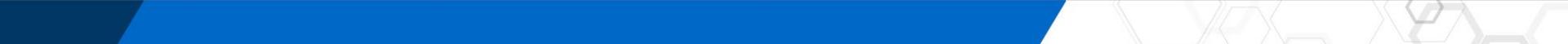
- <https://kitware.github.io/trame/>
- <https://trame.readthedocs.io/>

## ◆ Tutorial & Examples

- <https://github.com/Kitware/trame-tutorial>
- <https://github.com/Kitware/trame> => ./examples

## ◆ Cookiecutter

- <https://github.com/Kitware/trame-cookiecutter>



# Trame and VTK



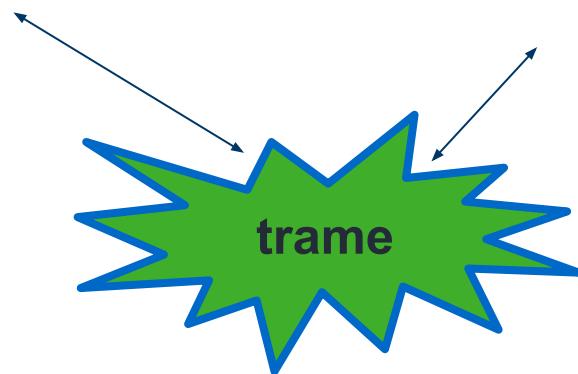
50 minutes

## VTK (C++, Python)

- ◆ Server only (python)
- ◆ I/O
- ◆ Data processing
- ◆ Rendering

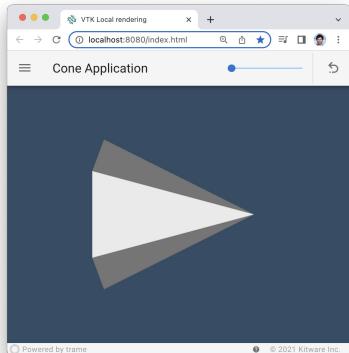
## Vtk.js

- ◆ Client only (browser)
- ◆ Rendering
- ◆ Limited processing
- ◆ Limited I/O



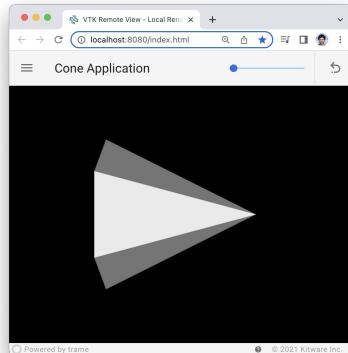
# How many ways can you make the same application?

No VTK, just vtk.js



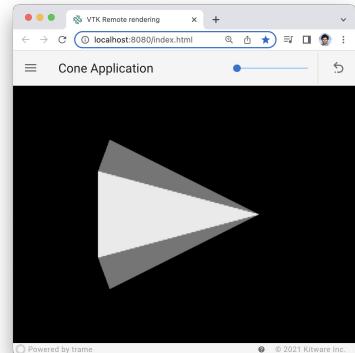
~ClientView.py

VTK for processing and  
vtk.js for rendering



LocalRendering.py

VTK for processing  
and rendering



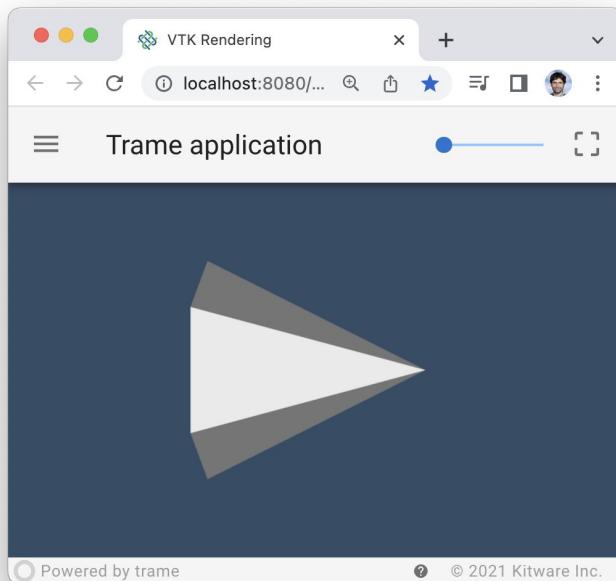
RemoteRendering.py

# Let's get started on your machine

```
mkdir trame-course && cd $_
python3.9 -m venv .venv
source ./venv/bin/activate
python -m pip install --upgrade pip
pip install "trame"

# Just to get the examples
git clone https://github.com/Kitware/trame.git
```

# VTK without VTK

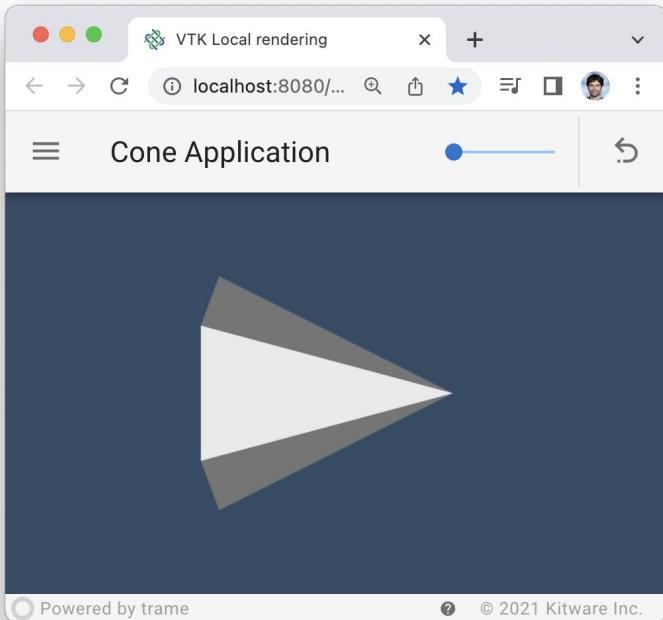


```
client.py  
LICENSE UPGRADE REQUIRED  
1 from trame.app import get_server  
2 from trame.ui.vuetify import SinglePageLayout  
3 from trame.widgets import vuetify, vtk  
4  
5 server = get_server()  
6 state, ctrl = server.state, server.controller  
7 state.trame_title = "VTK Rendering"  
8  
9 with SinglePageLayout(server) as layout:  
10     with layout.content:  
11         with vuetify.VContainer(fluid=True, classes="pa-0 fill-height"):  
12             with vtk.VtkView(ref="view"):  
13                 with vtk.VtkGeometryRepresentation():  
14                     vtk.VtkAlgorithm(  
15                         vtkClass="vtkConeSource", state={"resolution":}  
16                     )  
17             with layout.toolbar:  
18                 vuetify.VSpacer()  
19                 vuetify.VSlider(  
20                     hide_details=True,  
21                     v_model=("resolution", 6),  
22                     min=3, max=60, step=1,  
23                     style="max-width: 300px;",  
24                 )  
25                 with vuetify.VBtn(icon=True, click="$refs.view.resetCamera()):  
26                     vuetify.VIcon("mdi-crop-free")  
27  
28  
29  
30 if __name__ == "__main__":  
31     server.start()  
32
```

Line 32, Column 1      Spaces: 4      Python

```
python ./trame/examples/06_vtk/00_ClientOnly/client-side-cone.py
```

# VTK mesh rendering

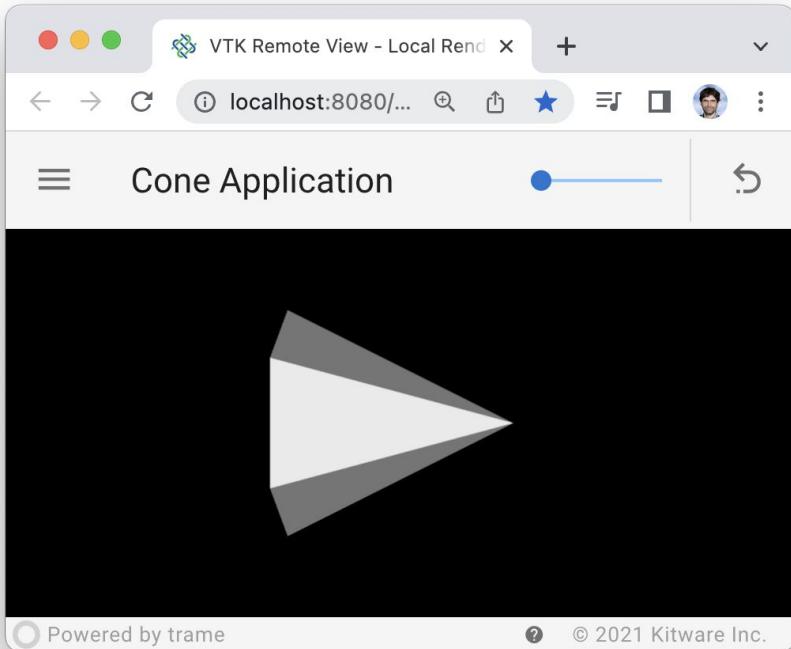


```
client-servermesh.py
LICENSE UPGRADE REQUIRED

1 from trame.app import get_server
2 from trame.widgets import vuetyfy, vtk
3 from trame.ui.vuetyfy import SinglePageLayout
4
5 from vtkmodules.vtkFiltersSources import vtkConeSource
6
7 server = get_server()
8 state, ctrl = server.state, server.controller
9 state.trame_title = "VTK Local rendering"
10
11 DEFAULT_RESOLUTION = 6
12 cone_generator = vtkConeSource()
13
14 @state.change("resolution")
15 def update_cone(resolution=DEFAULT_RESOLUTION, **kwargs):
16     cone_generator.SetResolution(resolution)
17     ctrl.mesh_update()
18
19 def update_reset_resolution():
20     state.resolution = DEFAULT_RESOLUTION
21
22 with SinglePageLayout(server) as layout:
23     layout.icon.click = ctrl.view_reset_camera
24     layout.title.set_text("Cone Application")
25
26     with layout.toolbar:
27         vuetyfy.VSpacer()
28         vuetyfy.VSlider(
29             v_model=("resolution", DEFAULT_RESOLUTION),
30             min=3, max=60, step=1,
31             hide_details=True, dense=True,
32             style="max-width: 300px",
33         )
34         vuetyfy.VDivider(vertical=True, classes="mx-2")
35
36         with vuetyfy.VBtn(icon=True, click=update_reset_resolution):
37             vuetyfy.VIcon("mdi-undo-variant")
38
39     with layout.content:
40         with vuetyfy.VContainer(fluid=True, classes="pa-0 fill-height"):
41             with vtk.VtkView() as view:
42                 ctrl.view_reset_camera = view.reset_camera
43                 with vtk.VtkGeometryRepresentation():
44                     html_polydata = vtk.VtkPolyData("cone", dataset=cone_generator)
45                     ctrl.mesh_update = html_polydata.update
46
47 if __name__ == "__main__":
48     server.start()
```

```
pip install vtk
python ./trame/examples/06_vtk/01_SimpleCone/ClientView.py
```

# VTK rendering without rendering



```
from vtk import *
from trame.app import get_server
from trame.widgets import vueify, vtk
from trame.ui.vueify import SinglePageLayout

server = get_server()
state, ctrl = server.state, server.controller
DEFAULT_RESOLUTION = 6

renderer = vtkRenderer()
renderWindow = vtkRenderWindow()
renderWindow.AddRenderer(renderer)
renderWindowInteractor = vtkRenderWindowInteractor()
renderWindowInteractor.SetRenderWindow(renderWindow)
renderWindowInteractor.GetInteractorStyle().SetCurrentStyleToTrackballCamera()

cone_source = vtkConeSource()
mapper = vtkPolyDataMapper()
actor = vtkActor()
mapper.SetInputConnection(cone_source.GetOutputPort())
actor.SetMapper(mapper)
renderer.AddActor(actor)
renderer.ResetCamera()
renderWindow.Render()

@state.change("resolution")
def update_cone(resolution=DEFAULT_RESOLUTION, **kwargs):
    cone_source.SetResolution(resolution)
    ctrl.view_update()

def update_reset_resolution():
    state.resolution = DEFAULT_RESOLUTION

with SinglePageLayout(server) as layout:
    layout.icon.click = ctrl.view_reset_camera

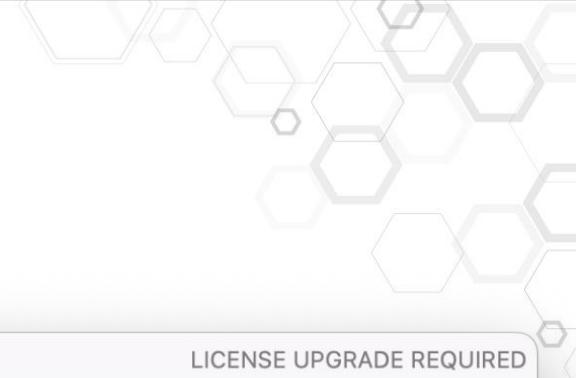
    with layout.toolbar:
        vueify.VSpacer()
        vueify.VSlider(
            v_model=("resolution", DEFAULT_RESOLUTION),
            min=3, max=60, step=1,
            hide_details=True, dense=True, style="max-width: 300px",
        )
        vueify.VDivider(vertical=True, classes="mx-2")
        with vueify.VBtn(icon=True, click=update_reset_resolution):
            vueify.VIcon("mdi-undo-variant")

    with layout.content:
        with vueify.VContainer(fluid=True, classes="pa-0 fill-height"):
            view = vtk.VtkLocalView(renderWindow, ref="view")
            ctrl.view_update = view.update
            ctrl.view_reset_camera = view.reset_camera

if __name__ == "__main__":
    server.start()
```

```
python ./trame/examples/06_vtk/01_SimpleCone/LocalRendering.py
```

# VTK rendering with rendering

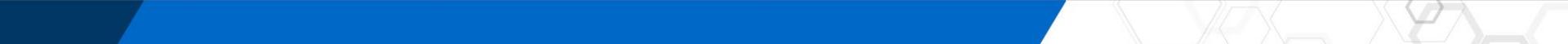


```
vtk-localrendering.py -> vtk-remoterendering.py — Desktop LICENSE UPGRADE REQUIRED  
1 with layout.content:  
2     with vuety.VContainer(fluid=True, classes="pa-0 fill-height"):  
3 -         view = vtk.VtkLocalView(renderWindow, ref="view")  
4 +         view = vtk.VtkRemoteView(renderWindow, ref="view")  
5             ctrl.view_update = view.update  
6             ctrl.view_reset_camera = view.reset_camera  
7  
Line 7, Column 2 29% Tab Size: 4 Diff
```

```
python ./trame/examples/06_vtk/01_SimpleCone/RemoteRendering.py
```

# What you've seen?

- ➊ **Layout (`trame.ui.*`)**
- ➋ **Widgets (`trame.widgets.*`)**
- ➌ **Trame server, state, controller (`trame.app.get_server`)**
  - State usage, initial value, monitoring changes
  - Method binding with UI and/or controller
- ➍ **VTK can be integrated with various granularity**
  - `vtk.js` only
  - VTK mesh and `vtk.js` rendering
  - VTK rendering with either local or remote rendering



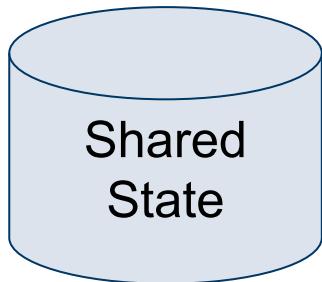
# Cheat sheets



10 minutes

# State handling

```
from trame.app import get_server  
server = get_server()  
state = server.state
```



```
@state.change("a", "b", "c")  
def change_detected(a, b, c, d, **kwargs):  
    pass  
  
state.a = 10  
print(f"a={state.a}")  
  
def update_var(name, value)  
    print(f"Before: {name}={state[name]}")  
    state[name] = value  
    print(f"After: {name}={state[name]}")
```

```
Widget(key="welcome")  
Widget(key=("welcome",))  
Widget(key=("welcome", "Hello"))
```

JS

- : Use string 'welcome'
- : Evaluate expression 'welcome'
- : Evaluate expression 'welcome' and initialize it to "Hello"

# Method handling and controller helper

```
from trame.app import get_server  
server = get_server()  
ctrl = server.controller
```

```
# @ctrl.set("hello_alias")  
def hello(*args, **kwargs):  
    print(args, kwargs)  
  
@ctrl.add("hello_alias")  
def hello2(*args, **kwargs):  
    print("v2:", args, kwargs)  
  
ctrl.hello_alias = hello  
ctrl.hello_alias('arg_0', key='yes')
```

Method calls

JS

```
VBtn(click=fn) : Call the function with no args  
VBtn(click=(fn, "[1, $event, 'hello']")) : Call the function like fn(1, event, "hello")  
VBtn(click=(fn, "[{$event}]", "{ x: 5 }")) : Call the function like fn(event, x=5)
```

# From JavaScript to Python

```
<v-tooltip bottom>
  <template v-slot:activator="{ on, attrs }">
    <v-btn
      color="primary"
      dark
      v-bind="attrs"
      v-on="on"
    >
      Button
    </v-btn>
  </template>
  <span>Tooltip</span>
</v-tooltip>
```

```
from trame.widgets import html, vuety
with vuety.VTooltip(bottom=True) as debug:
    with vuety.Template(v_slot_activator="{on, attrs}"):
        vuety.VBtn(
            "Button",
            color="primary",
            dark=True,
            v_bind="attrs",
            v_on="on",
        )
    html.Span("Tooltip")
print(debug.html) # To validate that the template match
```

Any invalid Python character become “\_” for any given attribute key.

# Cookiecutter

25 minutes

# Making your own application (Cookiecutter)

- ◆ Start with the cookiecutter

```
pip install cookiecutter  
cookiecutter gh:kitware/trame-cookiecutter
```

# Anatomy of the cookiecutter

- ◆ **Generic trame application**
  - Code layout and split
- ◆ **Bundles**
  - Desktop
  - Service with docker
- ◆ **Jupyter integration**
  - Embed application inside a cell
- ◆ **Example on how to bring your JavaScript widgets**
  - Adding vue.js components (option at creation)



20 minutes of exploration and experimentation

**Q&A**

```
project_name [Trame App]: Visualis
Select project_type:
1 - App
2 - App with Components
3 - Components
Choose from 1, 2, 3 [1]:
author [Trame Developer]: Kitware Inc.
short_description [An example Trame application]: VTK viewer for 3d stuff
Select license:
1 - BSD License
2 - MIT License
3 - ISC License (ISCL)
4 - Apache Software License
5 - GNU General Public License v3 (GPLv3)
6 - Other
Choose from 1, 2, 3, 4, 5, 6 [1]: 4
include_continuous_integration [y]: n
package_name [visualis]:
import_name [visualis]:
```

# Let's install it + various add-on...

```
cd visualis  
pip install -e .  
pip install pywebview jupyterlab
```

## Let's run it...

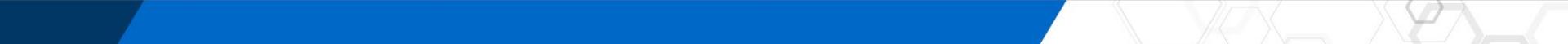
```
visualis
```

```
visualis --app
```

```
jupyter-lab
```

```
from visualis.app.jupyter import show  
show()
```

```
show(height=200)
```



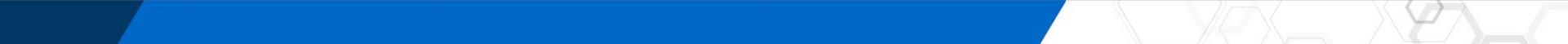
# ParaView



10 minutes

# ParaView Gotcha

- **ParaView can run VTK code too...**
  - We can use ParaView to provide VTK-EGL backend
- **ParaView has its own python but without trame...**
  - How to run your application with your venv?
    - Use “pvpython -m paraview.apps.trame ...”
    - Use a custom python start file
- **ParaView is not\* compatible with your python**
- **Try running paraview-visualizer**
  - <https://github.com/Kitware/paraview-visualizer>



# Deployment

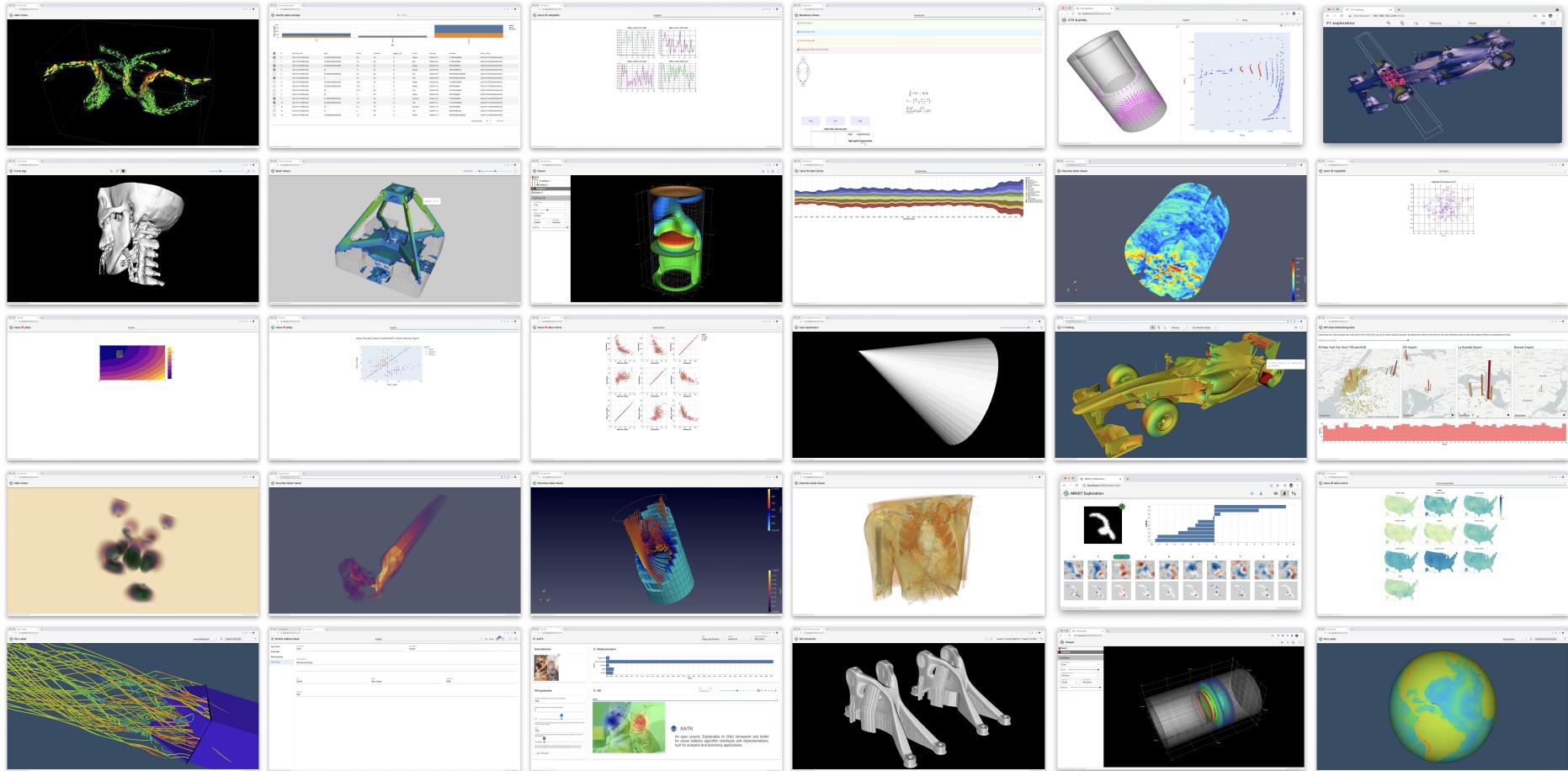


5 minutes

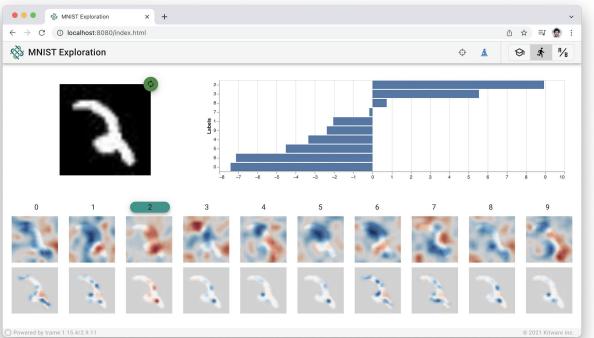
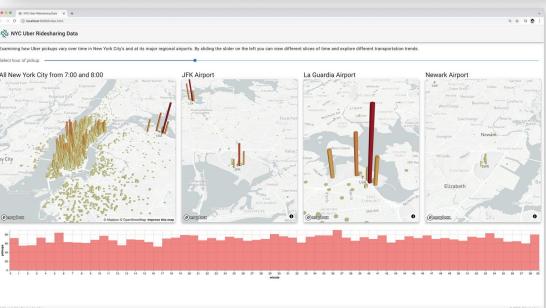
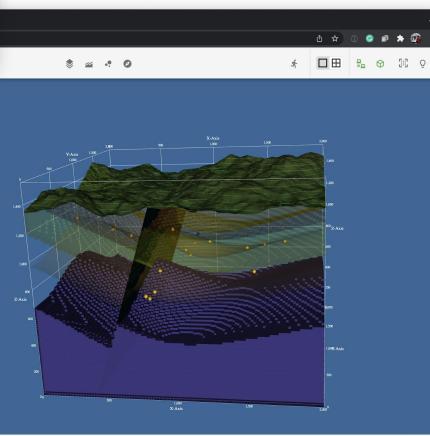
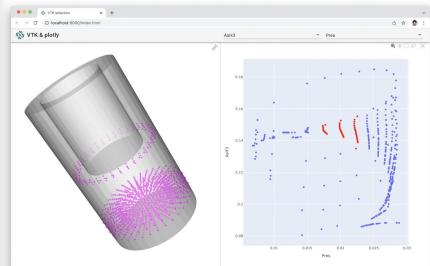
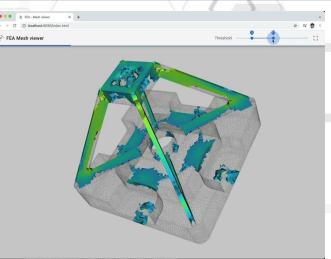
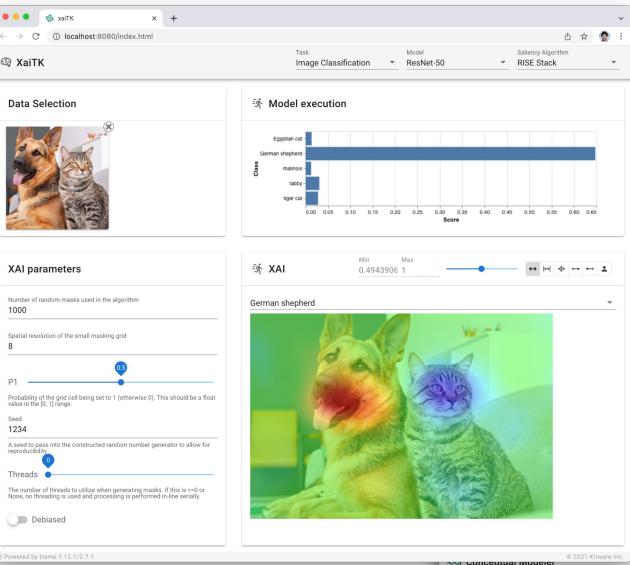
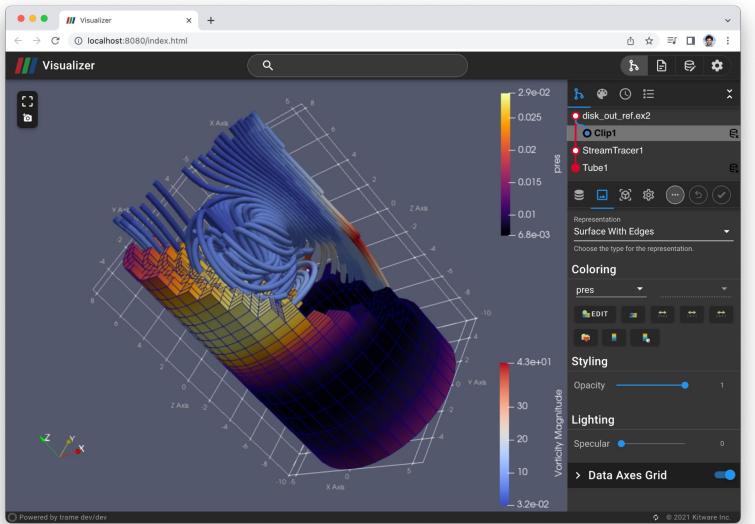
# Deployment

- **Desktop**
  - pywebview + pyinstaller
  - Use tk for file dialog and skip browser security limitation
- **Cloud service**
  - Docker image with multi users routing for a single node
  - Reverse connection + relay (infinity scaling)
- **Jupyter**
  - Trame provides helpers to run and display your application within your Jupyter cells

# Examples



# Applications



# Questions / Discussions

[sebastien.jourdain@kitware.com](mailto:sebastien.jourdain@kitware.com)

# Additional materials

- **Trame**

- <https://kitware.github.io/trame/>

- **Course video**

- <https://vimeo.com/761096621/af2287747f>

- **Course materials**

- <https://github.com/Kitware/trame-tutorial/tree/master/course/introduction>