sekoia

Meshroom
The cybersecurity mesh assistant

#OXA-granted-project #opensource #opencyberalliance

David Bizeul
Jérôme Fellus

All-in-one platform *vs* Cybersecurity Mesh architecture

All-in-one

- Unified operation model
- Unified UI/UX

- Captive Silo
- Expensive non-modular licensing
- Full replacement of existing stack
- Can’t cherry-pick functionalities

- Can’t be good at everything…

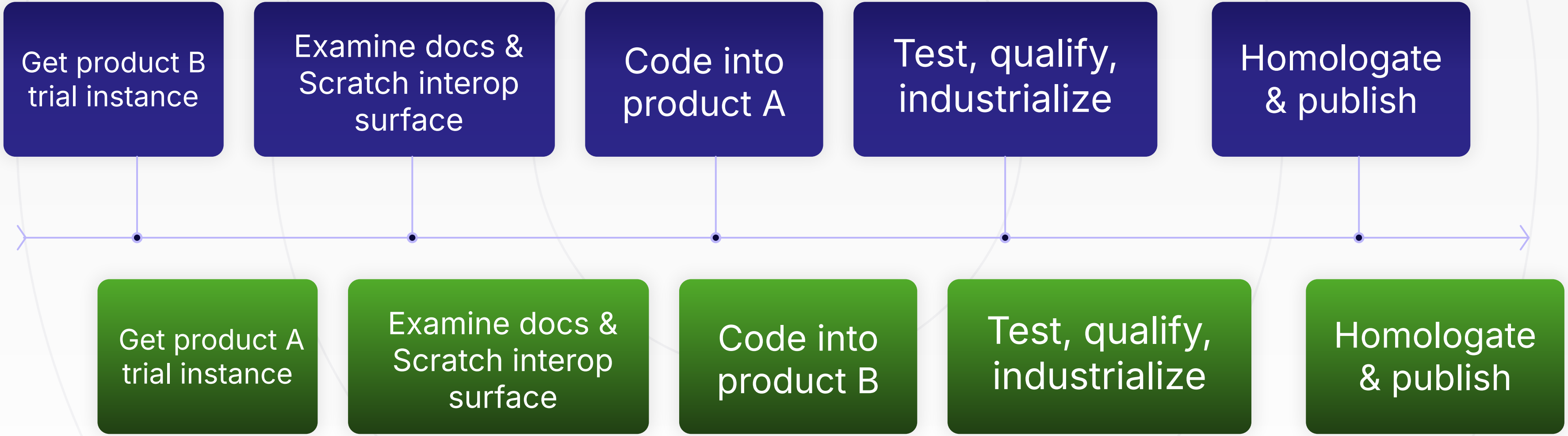
CSMA

- Favor interoperability
- Adapt & extend existing stack
- Do one job, do it right
- Focused expertise

- Need vendors cooperation
- Integration development burden
- Scattered SOC configuration

The N-to-N integrations curse

Product A



Get product B trial instance

Examine docs & Scratch interop surface

Code into product A

Test, qualify, industrialize

Homologate & publish

Get product A trial instance

Examine docs & Scratch interop surface

Code into product B

Test, qualify, industrialize

Homologate & publish

Product B

Building a mesh is ...

Cumbersome for vendors

Tedious for integrators

Unmanageable for devsec operators

Uncertain for buyers & end users

★ **Our contribution : an opensource assistant to compose cybersecurity meshes**

Compose…

Containerized stacks
docker compose up

Infrastructure-as-a-Service
terraform apply

Provisioning
ansible-playbook

→

Cybersecurity Mesh
📢 meshroom up !

Meshroom basic usage

```
meshroom init <path>

cd path

meshroom pull sekoia

meshroom create product

meshroom create integration
```

```
meshroom add

meshroom plug

meshroom up

meshroom produce

meshroom watch

meshroom down

meshroom publish
```

Meshroom model

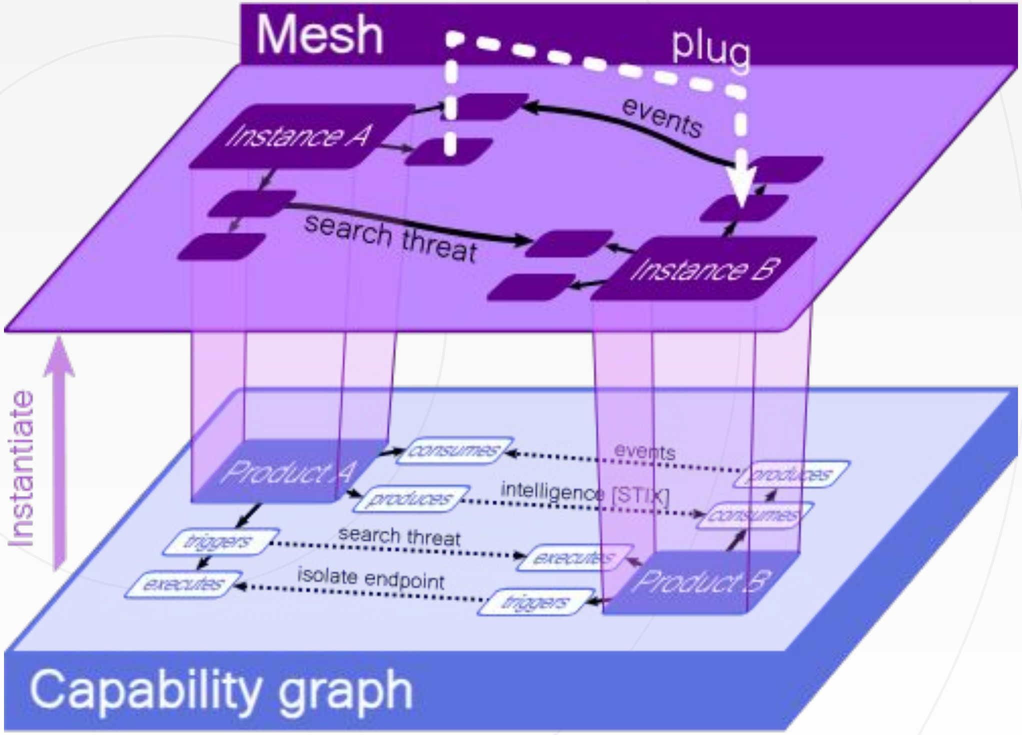
① Describe product capabilities

② Scaffold integrations between products

③ Instantiate products

④ Plug instances

⑤ meshroom up 🚀



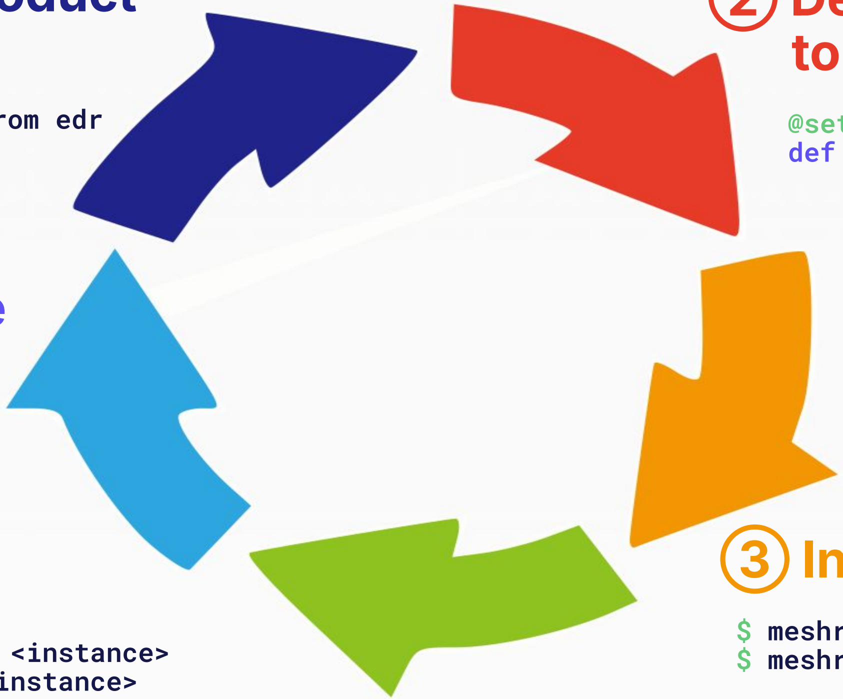
Assisted mesh integration journey

① Declare new product from template

```
$ meshroom create product –from edr
```

② Define python hooks to automate setup

```
@setup_consumer(‘events’)
def my_setup_func(plug: Plug):
    ...
```



③ Instantiate and plug

```
$ meshroom add <product> <name>
$ meshroom plug <instance> <instance>
```

④ meshroom up 🚀

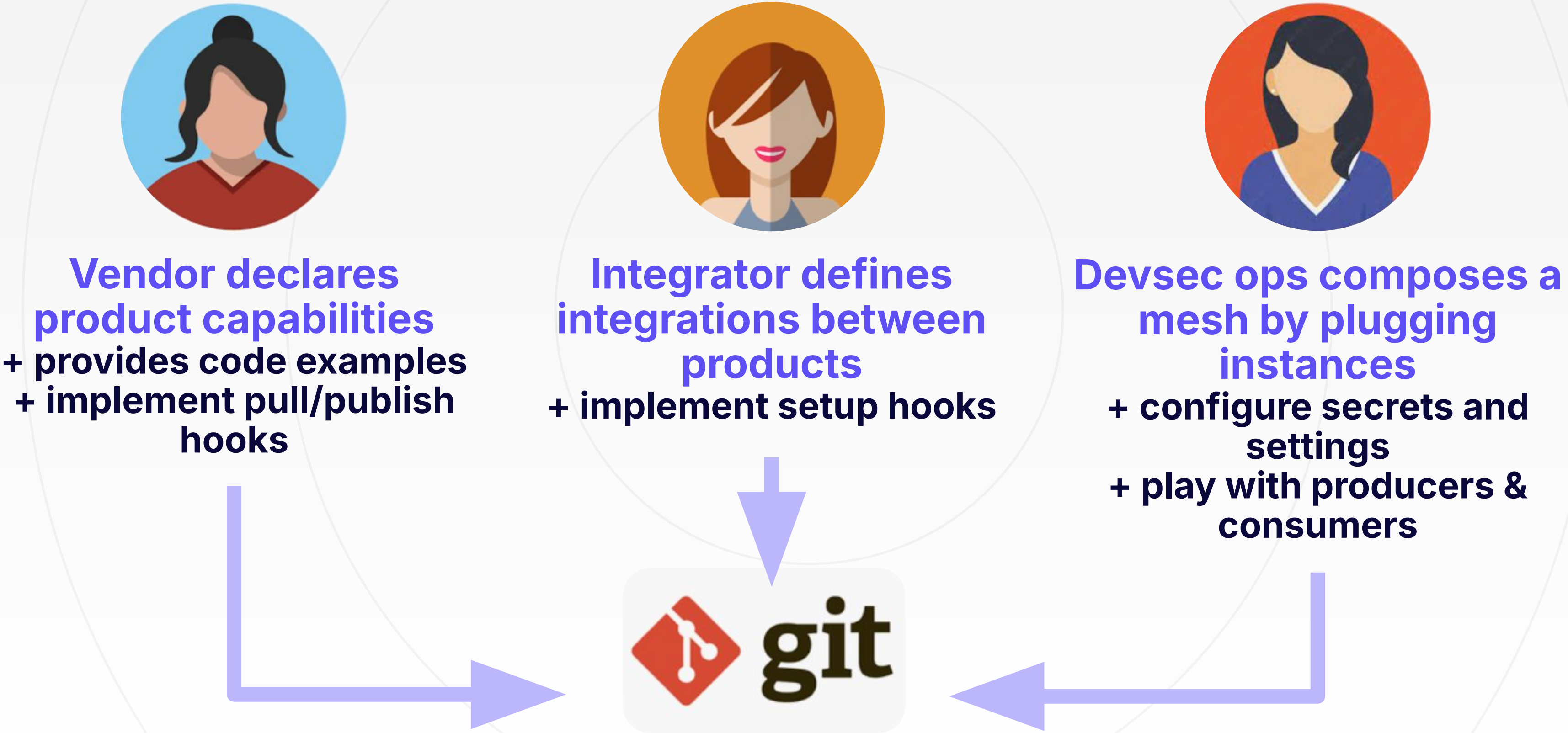
⑤ Play and test

```
$ meshroom produce <topic> <instance>
$ meshroom watch <topic> <instance>
```

⑥ Publish & share via git

```
$ meshroom publish <product>
```

Who ?



Vendor declares product capabilities
+ provides code examples
+ implement pull/publish hooks

Integrator defines integrations between products
+ implement setup hooks

Devsec ops composes a mesh by plugging instances
+ configure secrets and settings
+ play with producers & consumers

Everyone publish 🎉

How ?

- **producer→consumer**

  producer sends data to a topic,
  consumer receives data from the topic

- **trigger→executor**

  trigger submit commands to a topic,
  executor executes commands submitted to the topic

- **operation mode**

  push mode: source is active, destination is passive *(e.g., HTTP API)*
  pull mode: producer is passive, consumer is active *(e.g., syslog forwarding)*

- **plug ownership**

  cooperative: both producer & consumer need configuration to work *(e.g., AWS SQS)*
  unilateral: one end can setup everything without any action on the other end *(e.g., TAXII)*

- **python hooks**

  automate remote setup of real product instances and scaffolding of new integration via vendor-provided python functions executed upon meshroom commands *[see next slide 👀]*

📣 Full interop definition between two products

⦿ **Dataflow**
⦿ **Setup procedure**
⦿ **Boilerplate generator**

SEKOIA.IO - TLP GREEN

Hooks

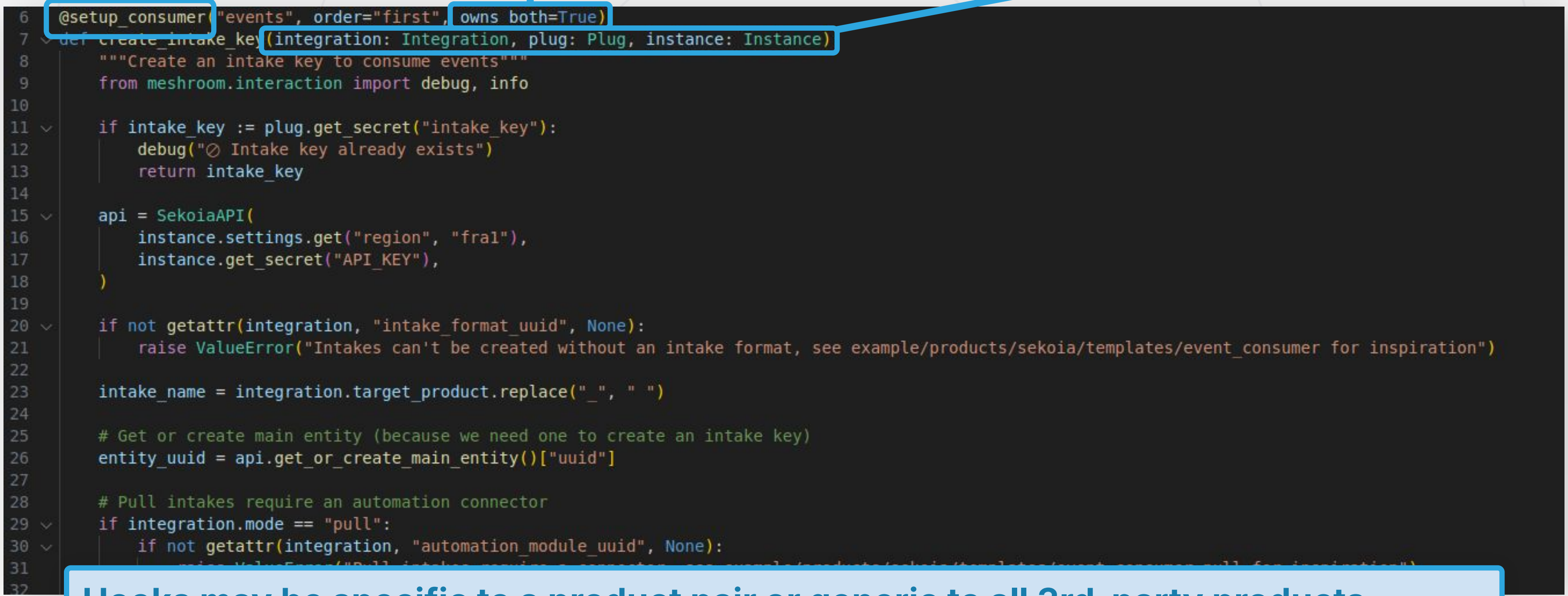
hook decorator	called upon	usage	
@setup	$ meshroom up	Define an automated setup step to get a plug up-and-running on a given instance	optional
@teardown	$ meshroom down	Define an automated step to shutdown and cleanup a plug from a given instance	optional
@scaffold	$ meshroom create integration	Generate files for a new integration for a certain topic	optional
@pull	$ meshroom pull	Generate integrations by pulling the vendor's online integration catalog	required
@publish	$ meshroom publish	Submit all defined integrations to the vendor's catalog for public homologation	required
@produce	$ meshroom produce	Send data to the plug's destination for testing	required
@watch	$ meshroom watch	Inspect data flowing through the plug	required

Hooks : example

Setup hook, called upon $ meshroom up

Unilateral setup
No remote configuration on producer side is required

Hooks have access to product instance and plugs



```python
@setup_consumer("events", order="first", owns_both=True)
def create_intake_key(integration: Integration, plug: Plug, instance: Instance):
    """Create an intake key to consume events"""
    from meshroom.interaction import debug, info

    if intake_key := plug.get_secret("intake_key"):
        debug("⊘ Intake key already exists")
        return intake_key

    api = SekoiaAPI(
        instance.settings.get("region", "fra1"),
        instance.get_secret("API_KEY"),
    )

    if not getattr(integration, "intake_format_uuid", None):
        raise ValueError("Intakes can't be created without an intake format, see example/products/sekoia/templates/event_consumer for inspiration")

    intake_name = integration.target_product.replace("_", " ")

    # Get or create main entity (because we need one to create an intake key)
    entity_uuid = api.get_or_create_main_entity()["uuid"]

    # Pull intakes require an automation connector
    if integration.mode == "pull":
        if not getattr(integration, "automation_module_uuid", None):
```

Hooks may be specific to a product pair or generic to all 3rd-party products

Meshroom features

- **Git-backed projects**
  For easy versioning and sharing

- **Builtin secrets store with GPG encryption**
  Keep all your instances' secrets in one secure place

- **One command to setup and teardown a full mesh**
  meshroom up / meshroom down

- **Scaffolding hooks**
  Help others building integrations with your products without pain

Going further ...

📣 **Meshroom is Github Copilot friendly**

Leverage code examples from your meshroom project
to bootstrap new integrations with LLMs

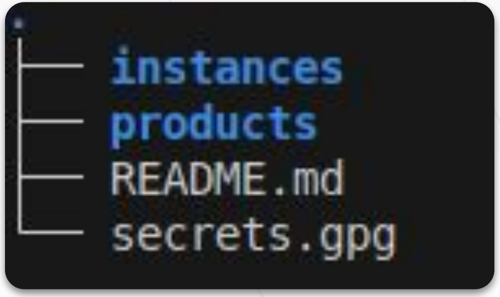
📣 **Let’s build the largest opensource mesh of cybersecurity products**

Contribute your meshroom materials to
https://github.com/opencybersecurityalliance/oxa

Step-by-step tutorial (1)

```
$ meshroom init <path>
```

- Initializes a git-backed meshroom project at *<path>*
- Creates the initial project structure



```
.
├── instances
├── products
├── README.md
└── secrets.gpg
```

- Starts with 0 product, 0 integration, 0 instance and 0 plug…

```
$ meshroom list products
```

```
$ meshroom list integrations
```

Step-by-step tutorial (2)

```
$ mkdir -p tmp
$ curl -L -o tmp.tar.gz https://github.com/jeromefellus-sekoia/meshroom/tarball/master
$ tar -xzf tmp.tar.gz -C tmp
$ mv tmp/*/example/products/sekoia products/sekoia
$ rm -rf tmp tmp.tar.gz
```

- Vendor has declared a product's capabilities and hooks
- Download specifications from a git repo
- We now have 1 product, with ready to use hooks. Let's use them !

```
$ meshroom pull sekoia
```

- @pull hook has downloaded all known integrations from Sekoia's official catalog

```
$ meshroom list products
```

```
$ meshroom list integrations
```

Step-by-step tutorial (3)

```
$ meshroom create product myedr --from edr
```

- Scaffolds a product definition from a predefined template of product capabilities [see *https://github.com/jeromefellus-sekoia/meshroom/tree/master/meshroom/templates/products*]
- We can define generic python hooks for our new product
  - @setup + @teardown hooks for $ meshroom up/down
  - @pull + @publish to grab and contribute to our product's official integrations catalog via $ meshroom pull/publish
  - @scaffold hook to provide code generators for $ meshroom create
  - @produce/@watch hooks for emulation via $ meshroom produce/watch

Step-by-step tutorial (4)

```
$ meshroom add sekoia mysekoia
$ meshroom add myedr
```

- Instantiate product instances
- Products may have defined settings and secrets : user is prompted for them here
- Nothing is submitted to the real user's tenants yet
- Instances are ready for calling `$ meshroom up`

```
$ meshroom list instances
```

Step-by-step tutorial (5)

```
$ meshroom plug events myedr mysekoia
$ meshroom plug search_threat mysekoia myedr
```

- Finds matching integrations
  - If one of the products has a unilateral setup hook [own_both=True], it takes ownership (no need for a defined integration on the other side)
  - Otherwise, find a pair of integrations matching the desired operation mode [push/pull] and topic
- Plugs instances to eachother
- Integrations may have defined settings and secrets: user is prompted for them here

```
$ meshroom list plugs
```

Step-by-step tutorial (6)

```
$ meshroom up 🚀
```

- Connect & configure each defined instance
- Execute @setup hooks to configure plugs
- Wait for the whole mesh to be ready

- You're now ready to use your Cybersecurity Mesh !

Step-by-step tutorial (7)

```
$ meshroom watch events myedr mysekoia
```

- Runs the @watch hook if defined on consumer side
- Inspects data flowing to the consumer and prints to standard output for debugging purposes

```
$ meshroom produce events myedr mysekoia
```

- Runs the @produce hook if defined on producer side
- Reads data from standard input and send it to the topic, as if it was produced by the producer itself

Step-by-step tutorial (8)

```
$ meshroom execute action myedr mysekoia
```

- Runs the @execute hook if defined on executor side
- Instructs the executor to directly execute the action as if it were sent by the trigger

```
$ meshroom trigger action myedr mysekoia
```

- Runs the @trigger hook if defined on trigger side
- Instructs the trigger to submit a command to its executor

Step-by-step tutorial (9)

```
$ meshroom down
```

- Cleanup all real product instances from what meshroom up had setup
- Leaves the user's tenants in a clean and predictable state

💡 $ meshroom up/down commands pair works exactly as
$ docker compose up/down commands pair

Step-by-step tutorial (10)

```
$ git commit -a -m “share my mesh” && git push
```

- Meshroom projects are git projects
  - ⦿ Use git to version your mesh
  - ⦿ Use git to share your mesh, privately or publicly
  - ⦿ Integrate contribution from other repos to extend your mesh

```
$ meshroom publish myedr
```

- Vendor can provide @publish hooks to streamline 3rd-party contributions to their integrations catalog sekoia

- 📣 https://github.com/jeromefellus-sekoia/meshroom
- 📣 https://jeromefellus-sekoia.github.io/meshroom/tutorial/