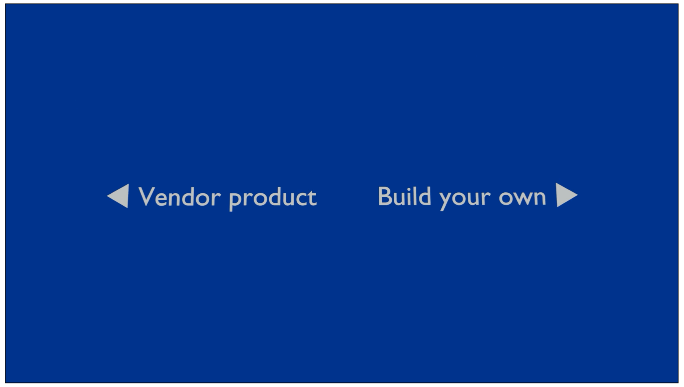
"hey anna, long time. where've you been?" "writing python"

I've worked for HEAnet forever, but the job's changed a lot. I spent a while doing the network stuff; then I project managed the big changes to network stuff; and for a while I was service desk manager so talking to clients about the stuff that was happening.

Then I disappeared for a bit. This is because I looked around, and i wanted to work on the stuff that helps us to deliver the network stuff. I saw the tools software developers had, and wanted to bring them to the network. Things like continuous integration, which lets us make more changes, more frequently, with lower risk than before.

[BUILD]

And an enormous part of that is how you provision, and de-provision, your equipment and your services. To do this, we have two problems to solve. The first problem to solve is: do you buy a provisioning and orchestration system from a vendor, or do you try to build your own?



◀ Vendor product Build your own ▶

* Where we found ourselves when we last made this journey in 2016. Replaced every piece of hardware in the network, ≈250 devices.

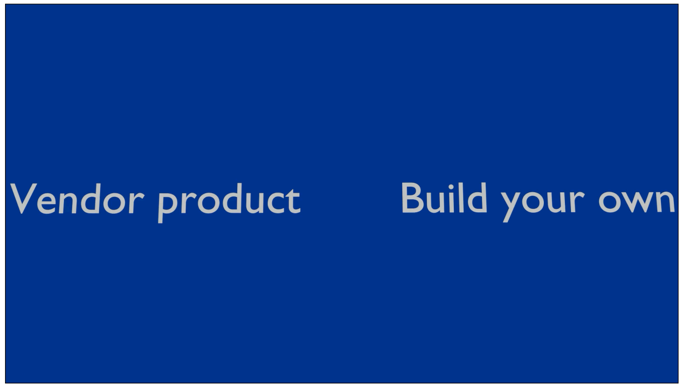
Bought an inventory system and an integrated service provisioning tool from the hardware vendor.

Worked well. But we spent lots of time customising the system to build services the way we wanted them, as fits our business.

And we had to build a whole separate application to integrate it with our DNS & monitoring systems

In 2023 we were notified that the tool we were using would be EOL by

2025. Ok, two years to plan.
Then a few months later we discovered that upgrading our router OS ... broke the provisioning tool.
What do we do?



Vendor product Build your own

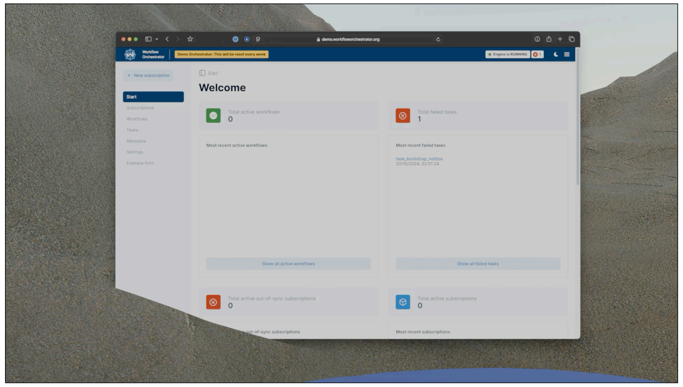
networks team in the meantime are getting by with Ansible playbooks and YAML files in git.
With a bit of CI, it works okay, but no one wants to be a Senior YAML engineer.
choices:
* implementing own solution
    * software is hard
    * software dev's not our org's primary expertise
    * and you're doing more of it every time you get a new vendor or device
* using vendor tools
    * very little cross platform
    * it's a treadmill, as we just found out - on someone else's timeline
    * $$$$$$$$ - quotes from €100k to well over half a million
        * AND YOU STILL HAVE TO SPEND LOTS OF TIME BUILDING ANYWAY

* software isn't THAT hard... is it?

Let's go climb a mountain
I'm Anna Wilson, and this is HEAnet's automation journey

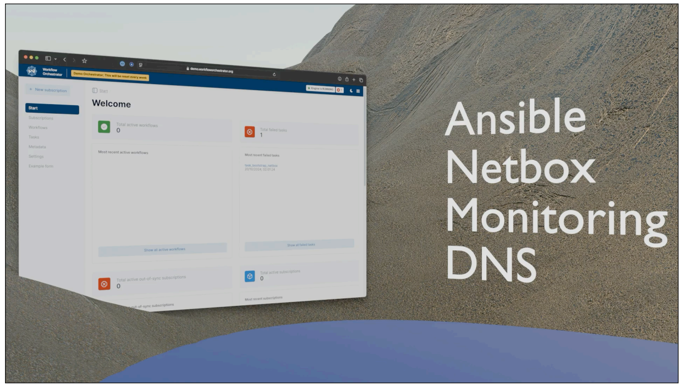
Something really important is that we're not starting from scratch.
We need an orchestrator, something that would let us manage all the parts of provisioning and deprovisioning in a coordinated way.

Last year we found that there is a really good open source product we could start with.
SURF, ESnet and GÉANT developed workflow orchestrator. decent fit. I'll get into its details in a moment, but the most important thing...



we need to work out our own products, and wire up our existing tools (e.g. ansible)
it's STILL a TON of work

and that's only the first problem
Remember I said there were two problems?
the second problem...



The unit of currency in HEAnet is the Project. That's how you measure time and get resources.

This is in our DNA; our whole history is in carefully deploying precisely specified infrastructure with static, inflexible amounts of money. this is what we're good at.

How do you manage a software project in this environment?

---

* To get there, let me tell you about the thing we're trying to build here

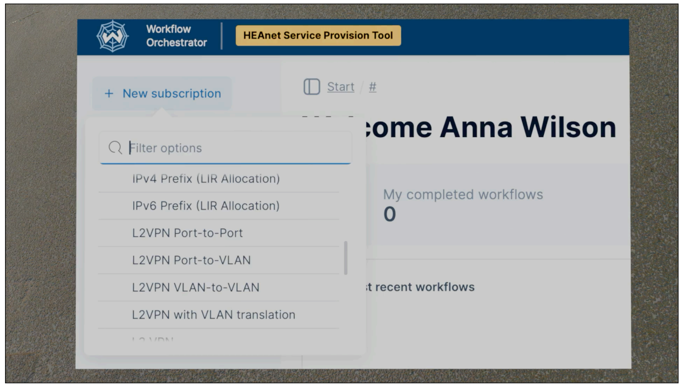
---

Workflow Orchestrator has the concept of products.

Things that you manipulate.

They can be nodes or ports, or things like point-to-point circuits, or even LIR assignments.

You go to create an instance of a product, you get a thing called a subscr iption. So let's do that.



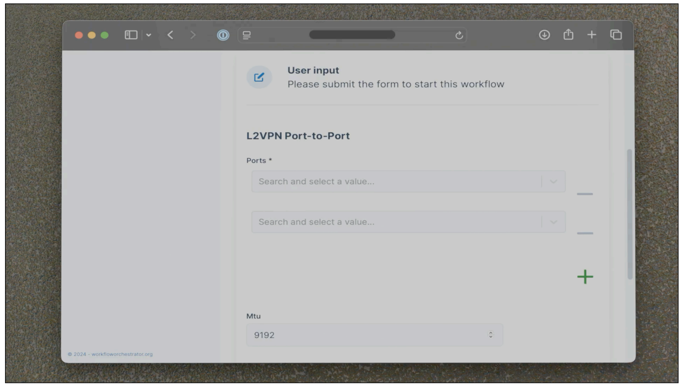
You go to create one of these, and you get a form with the details needed.

This is based on how you defined your product in the first place.

It's pulling in things like lists of valid ports from your inventory management system, or stuff like that.

In our case, each of these fields is fetching a list of available ports from Netbox, our single source of truth, and letting the user select one. So later we'll send this data back to Netbox to create a pseudowire, but in the meantime we're pulling data from it too.

This is the big shift from manually editing variables in text files or device configs. It's how we move away from the "Senior YAML Engineer" problem, which isn't just tedious, it's error prone.

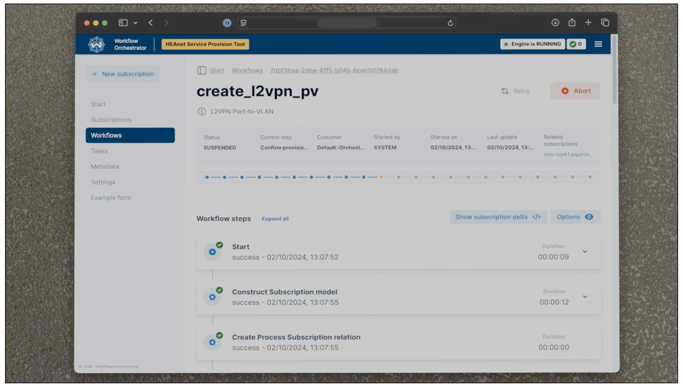

User input
Please submit the form to start this workflow
L2VPN Port-to-Port
Ports *
Search and select a value...
Search and select a value...
Mtu
9192

Then once it has all the info it needs, it kicks off the workflow.

How does that make changes?

WFO doesn't directly talk to your routers. If you're using NSO, or you're using ansible, or something else, you write a step for every action you need to happen, in a given order.

Nice green ticks when they've succeeded.



Start / Workflows / 7dbf3baa-2dba-41f5-b04b-bcec0d784dab

create_l2vpn_pv

L2VPN Port-to-VLAN

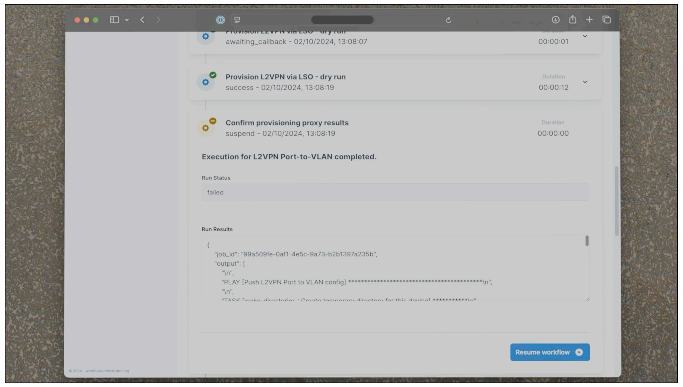
Status	Current step	Customer	Started by	Started on	Last update	Related subscriptions
SUSPENDED	Confirm provisio...	Default::Orchest...	SYSTEM	02/10/2024, 13:...	02/10/2024, 13:...	mtu-cork1.equinix...

Workflow steps

		Duration
Start	success - 02/10/2024, 13:07:52	00:00:09
Construct Subscription model	success - 02/10/2024, 13:07:55	00:00:12
Create Process Subscription relation	success - 02/10/2024, 13:07:55	00:00:00

If something goes wrong with a step, that's ok. Say your PowerDNS API isn't responding, or the zone doesn't exist yet or something.

Just like an engineer working manually, the workflow stops, you can troubleshoot, and then resume from the same step once that problem is solved



		Duration
	awaiting_callback - 02/10/2024, 13:08:07	00:00:01
Provision L2VPN via LSO - dry run	success - 02/10/2024, 13:08:19	00:00:12
Confirm provisioning proxy results	suspend - 02/10/2024, 13:08:19	00:00:00

Execution for L2VPN Port-to-VLAN completed.

Run Status

failed

Run Results

```
{
  "job_id": "99a509fe-0af1-4e5c-9a73-b2b1397a235b",
  "output": [
    "\n",
    "PLAY [Push L2VPN Port to VLAN config] *******************************************\n",
    "\n",
```

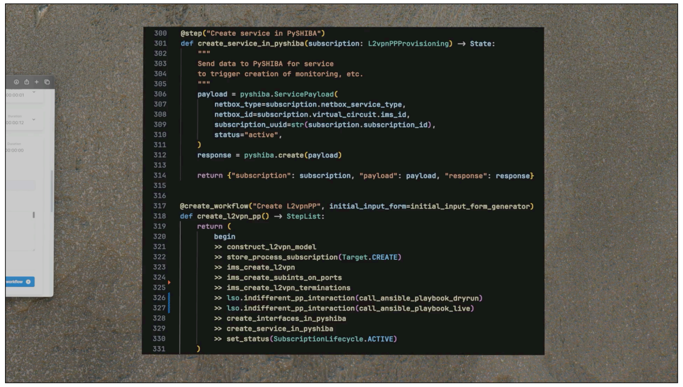
What is a step?

It's just a python function.

It takes the state of the subscription as input, performs whatever actions, and returns the updated state.

So most of what we're doing is designing our products, and then writing our workflows.

How do we keep track of that?



```python
@step("Create service in PySHIBA")
def create_service_in_pyshiba(subscription: L2vpnPPProvisioning) -> State:
    """
    Send data to PySHIBA for service
    to trigger creation of monitoring, etc.
    """
    payload = pyshiba.ServicePayload(
        netbox_type=subscription.netbox_service_type,
        netbox_id=subscription.virtual_circuit.ims_id,
        subscription_uuid=str(subscription.subscription_id),
        status="active",
    )
    response = pyshiba.create(payload)

    return {"subscription": subscription, "payload": payload, "response": response}


@create_workflow("Create L2vpnPP", initial_input_form=initial_input_form_generator)
def create_l2vpn_pp() -> StepList:
    return (
        begin
        >> construct_l2vpn_model
        >> store_process_subscription(Target.CREATE)
        >> ims_create_l2vpn
        >> ims_create_subints_on_ports
        >> ims_create_l2vpn_terminations
        >> lso.indifferent_pp_interaction(call_ansible_playbook_dryrun)
        >> lso.indifferent_pp_interaction(call_ansible_playbook_live)
        >> create_interfaces_in_pyshiba
        >> create_service_in_pyshiba
        >> set_status(SubscriptionLifecycle.ACTIVE)
    )
```

By making a lot of mistakes
Agile-ish approach still worked really well!

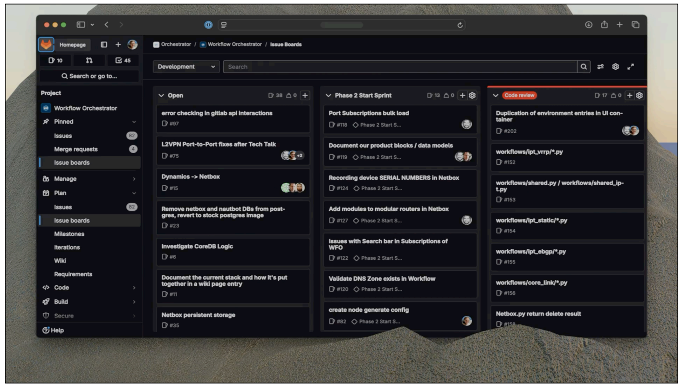
* We set up smart things at the start [screenshot: issue boards, merge requests]

* A non-dev could look at our issue boards and see the actual, live, state of the project

* Put effort into making docker etc work so we could all work on independent systems (this is non obvious)

* On a long scale, we have EXCELLENT visibility, and on a short scale, we

have enough detail to prioritise sensibly
* When we go the wrong direction, fairly quick to spot this and straightforward to course correct
...until we look outside our own bubble



Open 38 0

- error checking in gitlab api interactions #97
- L2VPN Port-to-Port fixes after Tech Talk #75
- Dynamics -> Netbox #15
- Remove netbox and nautbot DBs from post-gres, revert to stock postgres image #23
- Investigate CoreDB Logic #6
- Document the current stack and how it's put together in a wiki page entry #11
- Netbox persistent storage #35

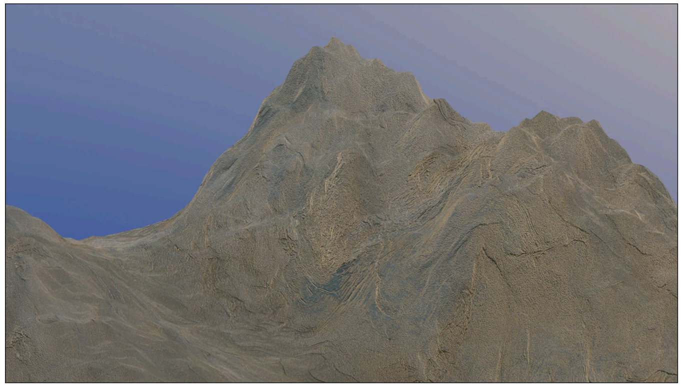
Phase 2 Start Sprint 13 0

- Port Subscriptions bulk load #118 Phase 2 Start S...
- Document our product blocks / data models #119 Phase 2 Start S...
- Recording device SERIAL NUMBERS in Netbox #124 Phase 2 Start S...
- Add modules to modular routers in Netbox #127 Phase 2 Start S...
- Issues with Search bar in Subscriptions of WFO #122 Phase 2 Start S...
- Validate DNS Zone exists in Workflow #120 Phase 2 Start S...
- create node generate config #82 Phase 2 Start S...

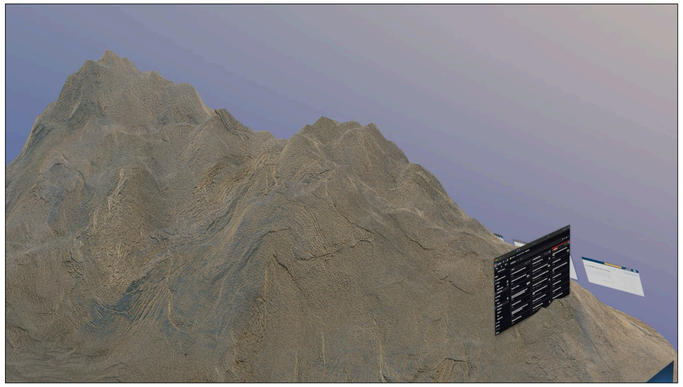
Code review 17 0

- Duplication of environment entries in UI con-tainer #202
- workflows/ipt_vrrp/*.py #152
- workflows/shared.py / workflows/shared_ip-t.py #153
- workflows/ipt_static/*.py #154
- workflows/ipt_ebgp/*.py #155
- workflows/core_link/*.py #156
- Netbox.py return delete result

But software dev is unpredictable (in both directions). This doesn't fit well in a time-bound project template.
I'll give you an example:
We were ready to deliver A Thing in April.
We knew we're not going to reach this peak...



...but how about THIS peak?
We said, we can't get you everything in the project initiation document in time, but we can at least get something deployed that the team could start using.



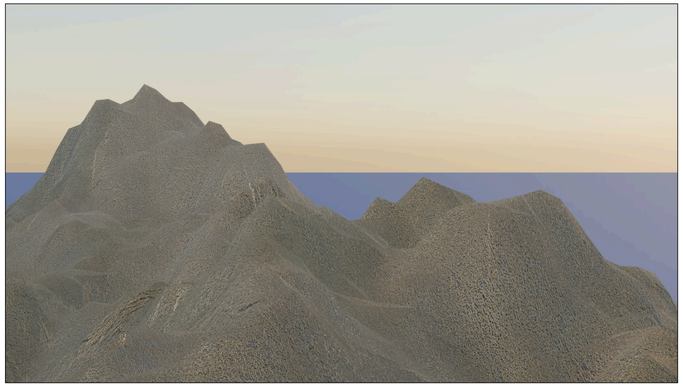
But our unit of currency is The Project. That's what gets you resources. When you finish the project, those resources go elsewhere.

I really mean that this is our DNA. We specify what we want, and then hold the supplier to precisely that. The two tools we have in public procurement are the spec at the beginning, and the project completion (and therefore paying the invoice) at the end.

Some years ago, I talked here about disruption theory, which is the idea that the very things that make organisations successful can also, in some circumstances, be a weakness.

In this case, this means that the organisational incentives work AGAINST us delivering in stages, and toward a "big bang" -- which increases risk

---



Focus on deliverable features subtly discourages spending time on fundamentals

I'll give you an example. We spent the first few months just getting our heads around the concepts, and setting up how we were going to work.

Our team is one developer with decent network experience, a few network engineers and former network engineers with varying amounts of python experience, and a project manager who knows networks inside out but hadn't managed a software project before.

The team spent three days in Utrecht with SURF and GÉANT and that gave us an enormous head start.

We spent some time on fundamentals. Someone had already set up a server with an instance of the software, but I'm sworn off running individual servers at this point. I want everything i run to be immutable images created from version control.

So we spent some time finding our rhythm with gitlab - branches, merge requests, continuous integration and deployment

docker was really important - took time to get everyone used to it, but what we're developing on our laptops is REALLY CLOSE to what gets deployed

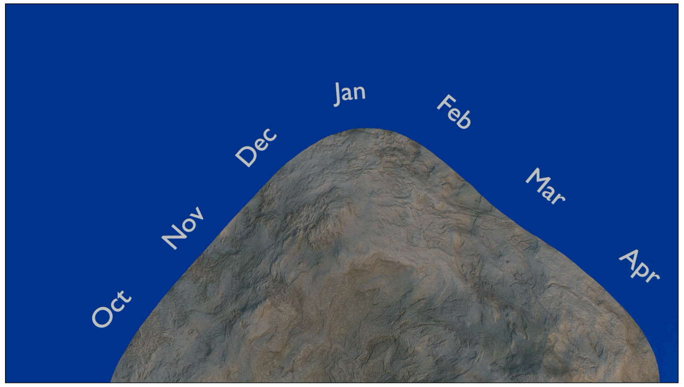
And we're also learning this software, its concepts, developing our picture of how we adapt it to our own environment. But from the outside, it looked like not much was happening.

Then we had what looked like a massive burst of activity from jan to apr; we had our basic fundamentals in place and could start working on features

We were still learning as we were going. unknown unknowns; we knew we were making mistakes that we would uncover later. thinking in an agile way - quick deployments, frequently re-evaluating priorities - but that wasn't giving the org what it wanted. what we've all been trained to work with.

So when we learned we wouldn't be commissioning in april, that also meant we didn't work on loading live data.

Why bother if the data and platform are both gonna change?



Project stretched on from Apr, into May, through the summer, then into the autumn...
and it's only the last few weeks we've started loading live data

But loading live data has been SUPER illuminating and would probably have overall saved time if we'd done it earlier

It's uncovered a LOT of those unknown unknowns and made concrete some abstract decisions we had trouble with

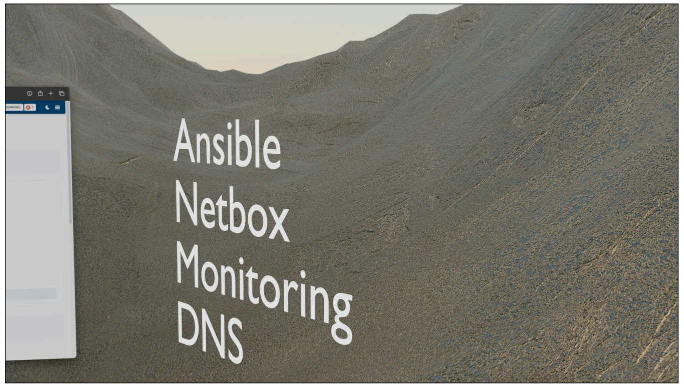
Like I said, our unit of currency is the project. I don't know what yours is. It's probably something very different. But I know for a fact that your organisation has one, and if you're having trouble effecting change, there's a good chance that this is why.

So, it's taken us a year. What have we got out of that?

What's clear to me now, in a way it wasn't before, is that this isn't just about wiring Ansible and Netbox together. This is encoding your business model in a tool

That's a big deal

An issue we're dealing with now is that our networks team's picture of our clients is subtly different to our finance team's picture. And that's maybe not a big deal day to day, but when two clients merge, or when a client changes name (and I know how important it is to get name changes right) - suddenly these departments need to work in concert in a way we didn't really have to before.



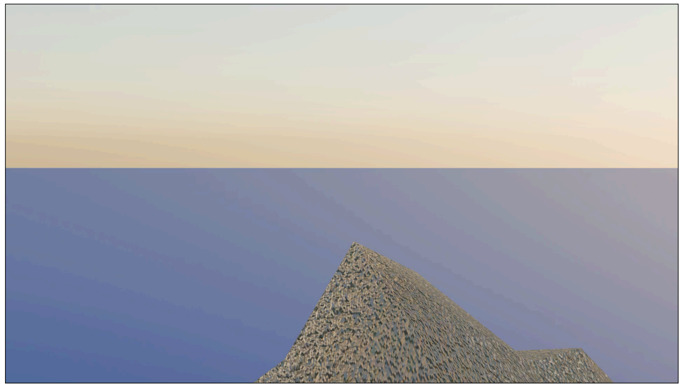
Ends up rewarding not just the network, but the whole org

We're touching network, service desk, finance

In some ways, we're finding out WHAT we know, coordinating our data between these departments

We're learning skills as an organisation about how to understand and implement our own business, not just faster, not just more consistently, but fundamentally better

We're even finding out things about the way that we do things, about what makes things legible across the organisation.



And if you can find out your unit of currency, your organisation can do this too

Thank you very much

Anna Wilson, HEAnet
Animated in Blender
Workflow Orchestrator workfloworchestrator.org
Pylons by neonbrand on Unsplash
Mountain texture from publicdomaintextures.com