

The Automation Archipelago

Red Hat
Summit

Connect

Connecting IT islands through collaboration,
standardization and automation at Fluvius

Arthur Odekerken — Expert Systemen en Operaties

Jurgen Van de Perre — DevOps Engineer



Topics

- 1 Fluvius — who we are
- 2 The challenges we face
- 3 Five islands — why this was hard
- 4 Collaboration, standardization, automation
- 5 What changed
- 6 What comes next

What Fluvius does

1 · Who we are



Electricity
138.501 km
3.7 million connection points



Natural gas
57.192 km
2.4 million connection points



Sewerage
12.489 km
703.105 connections



Heat
58 km
niche — 13 municipalities



Public lighting
1.21 M
light points, 74% already LED

Who does the work

~6.000

people

about 4.500

work in the field, not at a desk

26 hours

of training per person, every
year

82%

Great Place to Work score



Fluvius IT is very different from OT



IT runs the business

Requests, billing, the service catalogue, the workforce systems.

It changes every week. That is the point of it.

≠



OT runs the grid

Control, protection, measurement, the devices in the street.

It changes rarely, slowly, and under strict control.

A bad change in IT raises a ticket.

A bad change in OT causes an outage.

15 min

And the data volume only goes one way: 📶

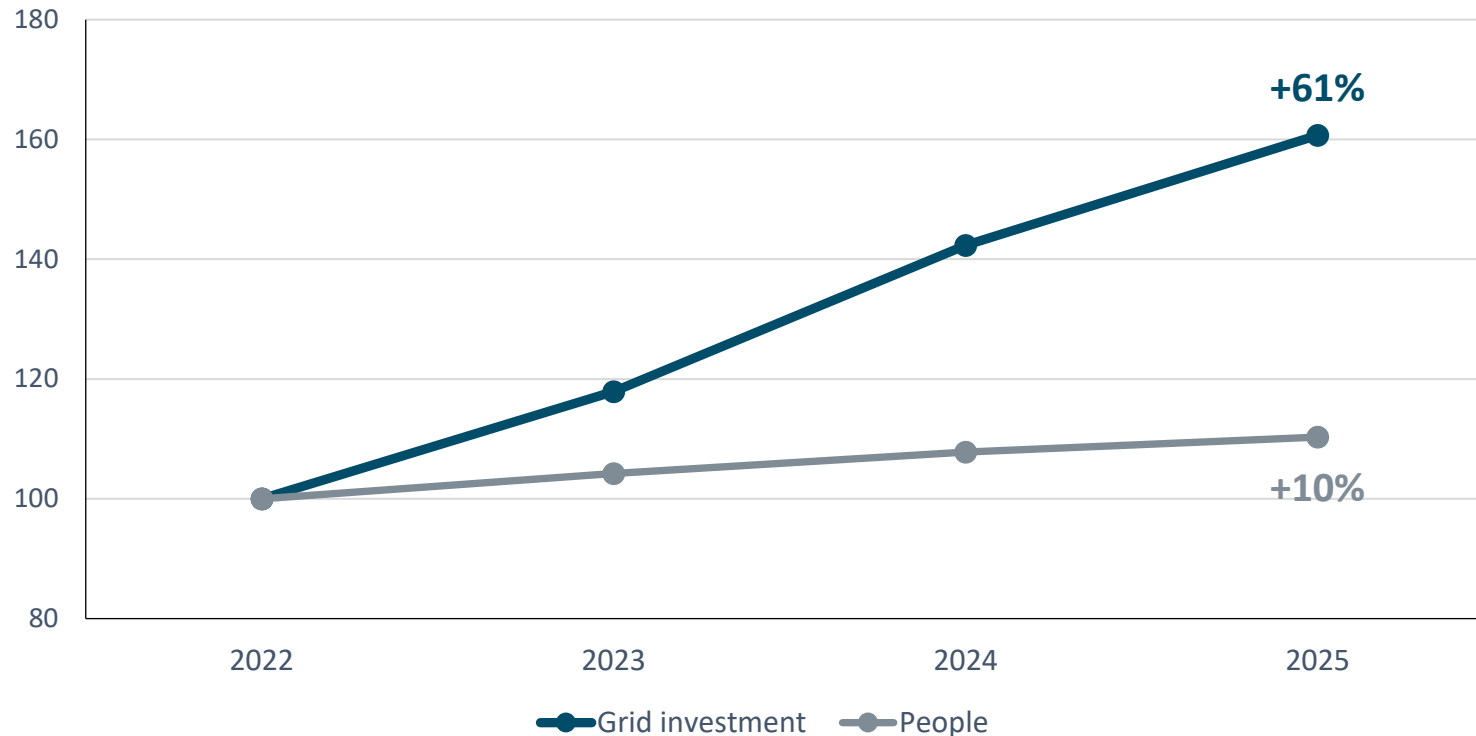
Since 1 January 2025, every digital electricity meter reports a value every 15 minutes.

The challenges we face

2 · The challenges we face



Investment in the grid keeps rising **+61%**. The team capacity is growing by only 10%.



6x

The work is growing six times faster than the team.

~€11 billion

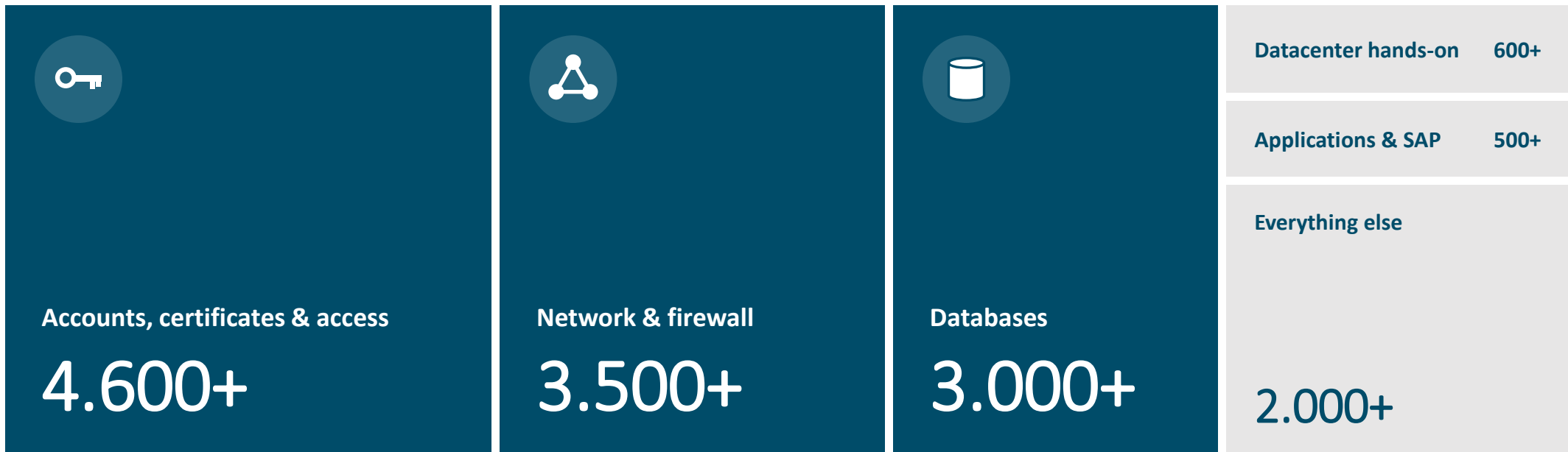
Already committed investments for 2026 – 2035.

The IT request catalogue



14.000+ requests, every one by hand

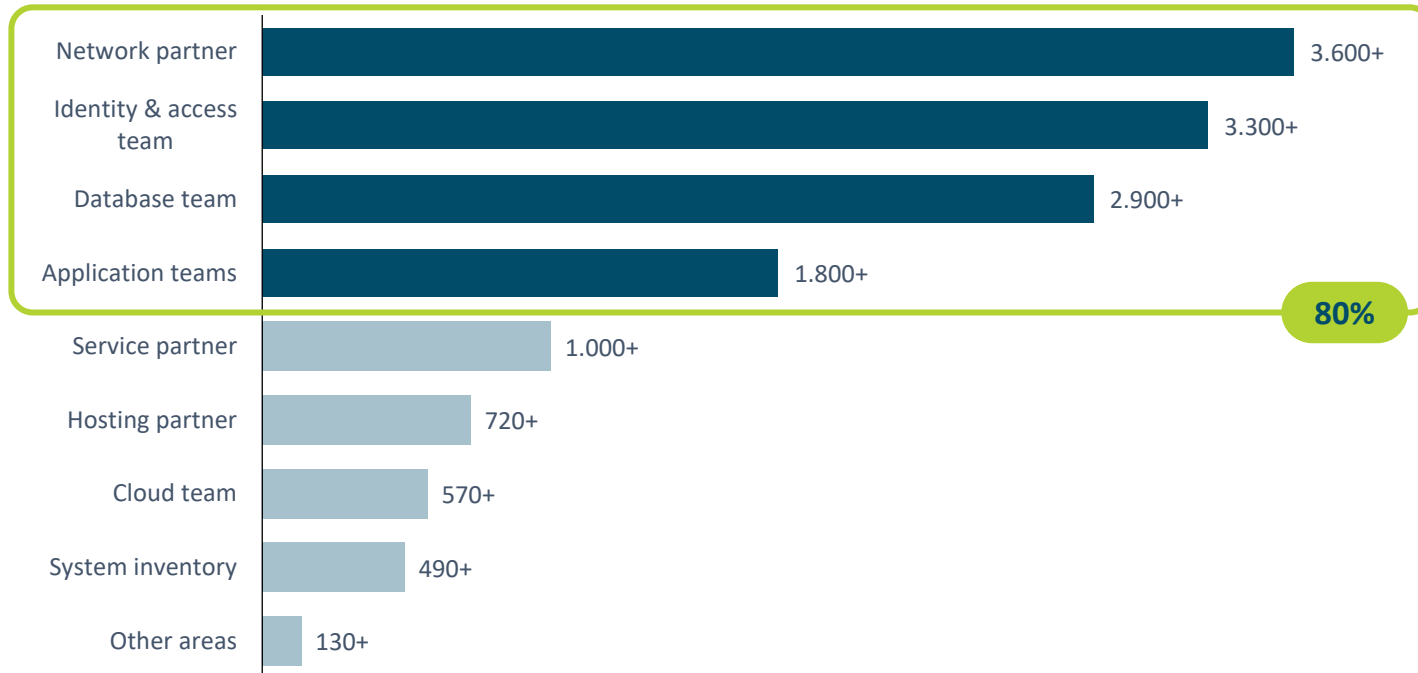
190 different request types



Four categories account for 80% of all requests.

IT requests by category

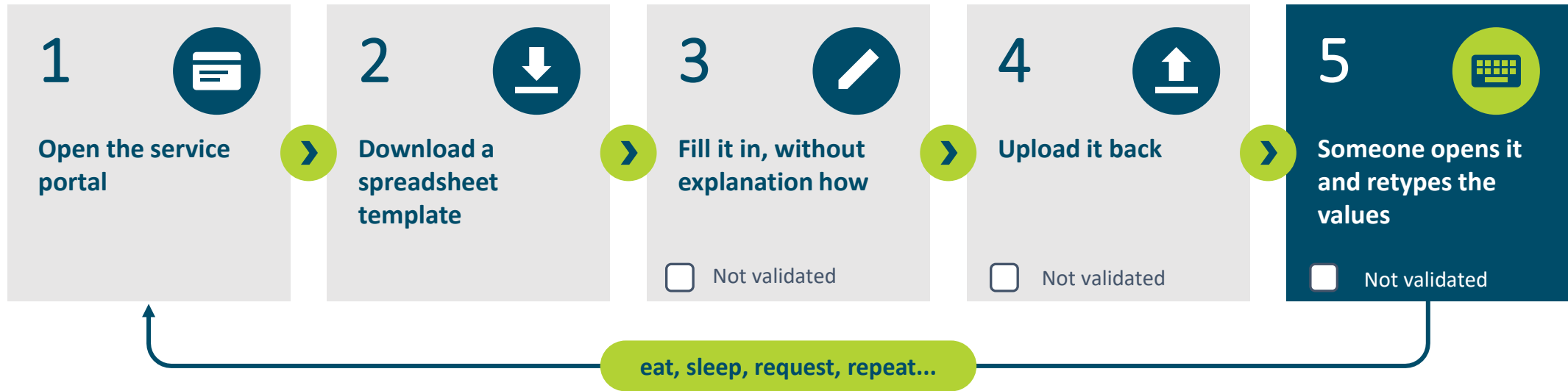
The same 14.000+ requests, by the area that handled them



Four areas carried 80% of it.



The journey of a request



Not one step validated anything.

Problems were often discovered days later, when somebody tried to use the service.

Every request was made artisanal



The spreadsheet checked nothing.

Not one field, not while you filled it in. If you were lucky the team caught it later, otherwise at test time, in production, or never.



The work was done twice.

Hard to check what already existed. Anywhere.



The result was incomplete.

5 business days for a server and still incomplete when delivered.

There was no single source of truth. There were 14.000+ of them.

Every team on its own island

No man is an island. Every team was.

5

3 · Five islands

1

1

The team with no processes

2

The team that can only check when live

3

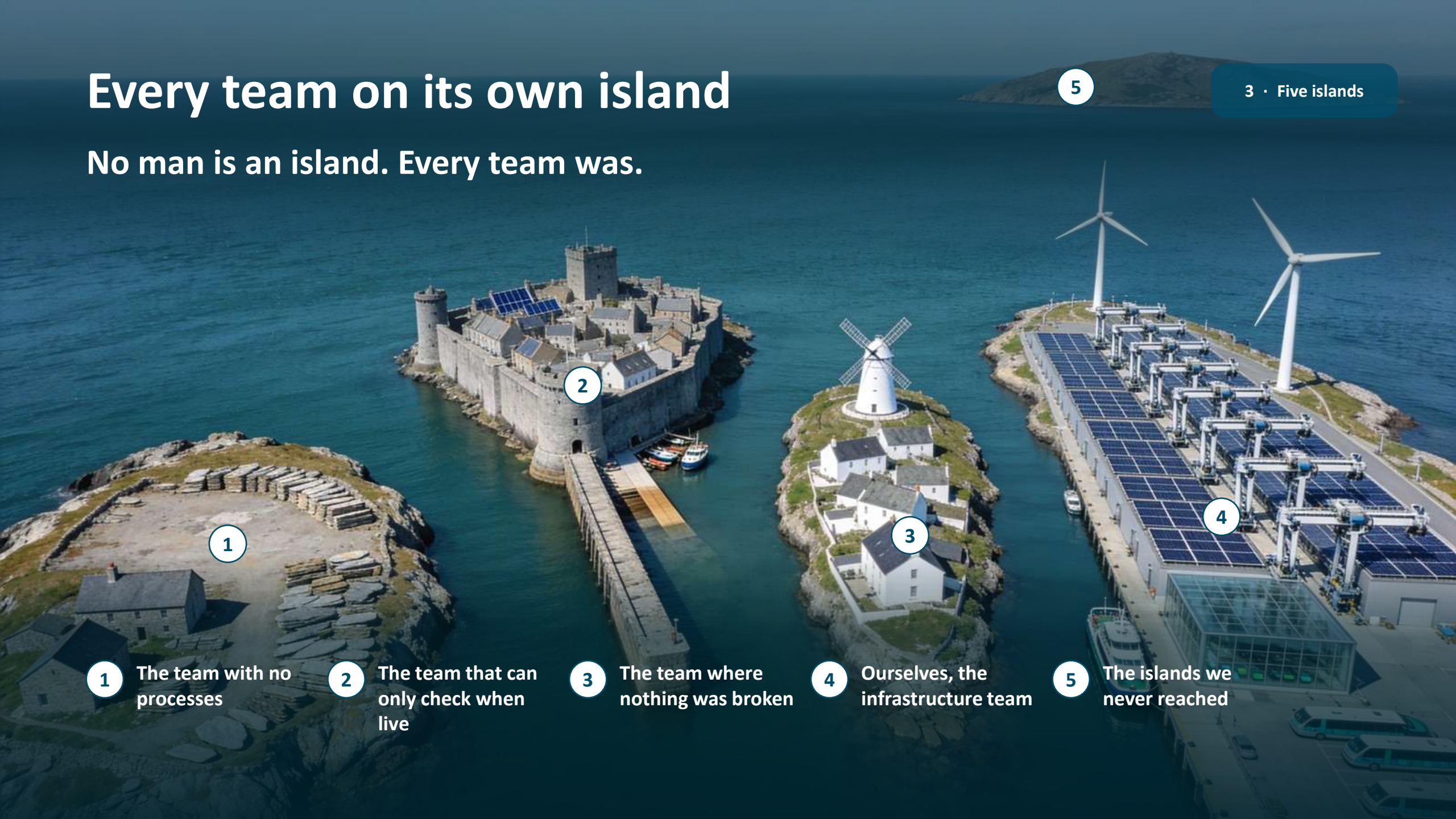
The team where nothing was broken

4

Ourselves, the infrastructure team

5

The islands we never reached



The team with no processes

3.300+

requests in 29 months,
none of them automated

You cannot protect them from
making mistakes

It depends who picks it up. A certificate took 5 to 7 days because of manual checks. Issuing one takes minutes.

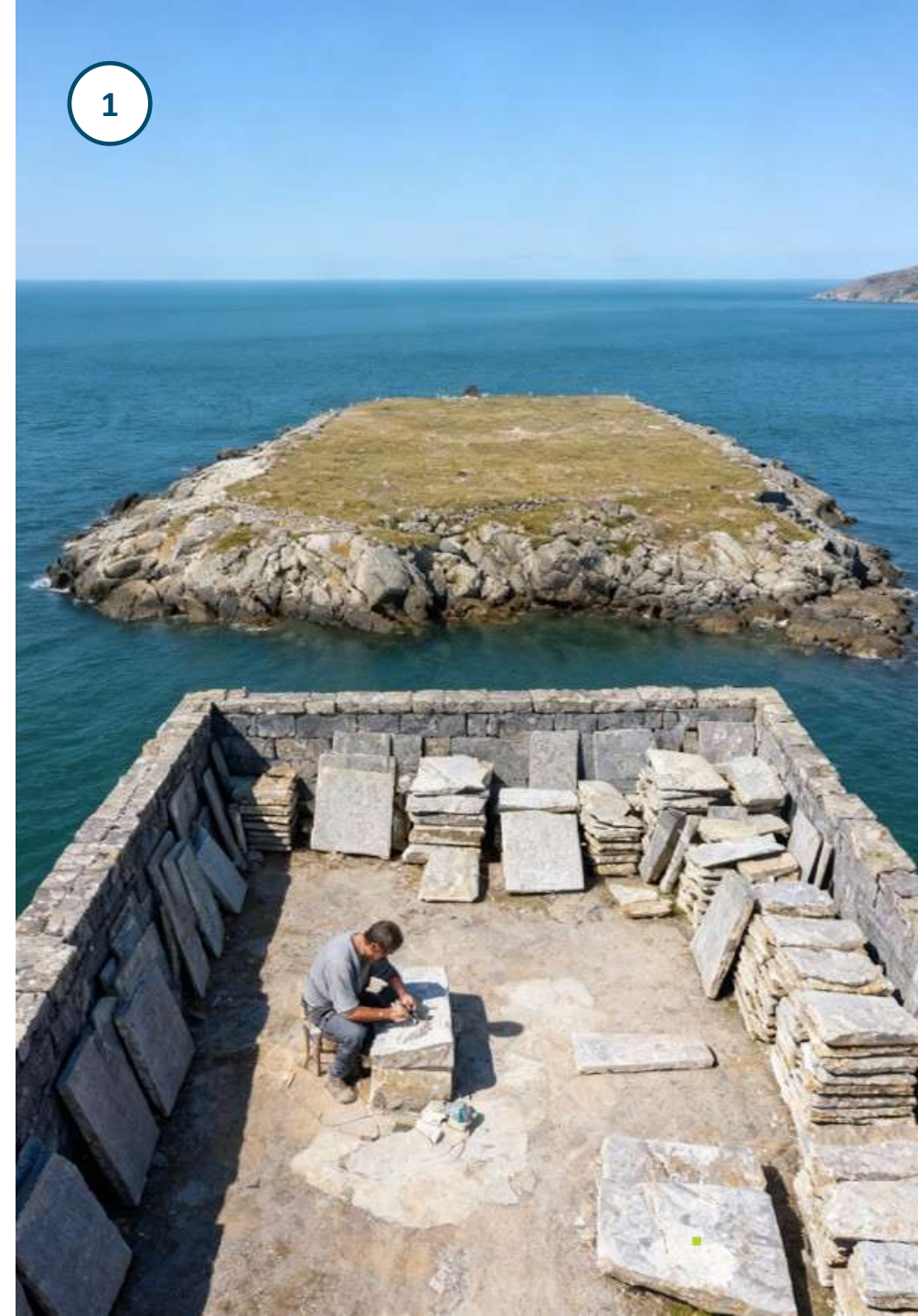
It depends on how they copy-paste. Account attributes arrived as free text. One wrong character and the account silently does not work.

It depends how they do it. We could not automate the process, because there weren't any documented internal procedures.

No processes means no consistency... and no automation

fluvius.

1





2

The team that can only check live

OTAP:
Oops, Test Always in Production

3.600+

requests in 29 months,
none of them automated

The environment is genuinely complex. Load balancers, software-defined fabric beside legacy switching, hub-and-spoke across datacenters, wireless controllers, access control appliances.

There is some automation, but no procedure. A six months inactive firewall rule is auto-removed, only a policy. The execution of the request depends on who does it.

Changes can only be validated in production. There is no dry run, find out if it works after the implementation, causes issues and downtime.

Not prepared, nothing repeatable to automate.

fluvius.

Changes were never without risks

2–3 days

for a single standard
change

60%

needed rework

The rework was not
because of the lack of
knowledge.

Requests arrived wrong.

Complex spreadsheets needed to be filled in and could not be validated during creation.

There was no test infrastructure.

No dry run means the only test is production, which is exactly where the 60% comes from.

And every change was adding to the fear.

A two-minute edit on a device nobody wanted to touch, waited for days until the SLA date.

Complexity created fear. Fear created delays.



Source: Fluvius automation team, internal measurement.

2



The team where nothing was broken

“

It works. Why would we change it?

1.800+

requests in 29 months,
none of them automated

They were not wrong. Custom scripts and procedures, written years ago and running on a fixed schedule everybody needs to follow.

Knowledge lives in people. Most of the knowledge resides with one person. Eventually that knowledge leaves with the person.

Nothing appeared broken. We arrived offering to solve a problem they did not have.

"If it ain't broke..." and honestly, it wasn't broken... yet

fluvius.

And us, the infrastructure island

5 days

to deliver a server,
and it arrived incomplete



Account



Hostname & IP



Server

One server. Three islands. Before it came back.

Our service partner raised the requests. The account from one island, the hostname and IP address from another. But only the server deployment was their responsibility.

They assembled it by hand. Waiting for every team to complete their steps, to eventually put the pieces together.

Updating the CMDB was difficult. Inventory and configuration mismatch, because every engineer typed it their own way.

A naming convention existed but nothing enforced it.



4



The islands we never reached

2.900+

requests in 29 months,
none of them automated

Nobody said no.
We simply never got there.

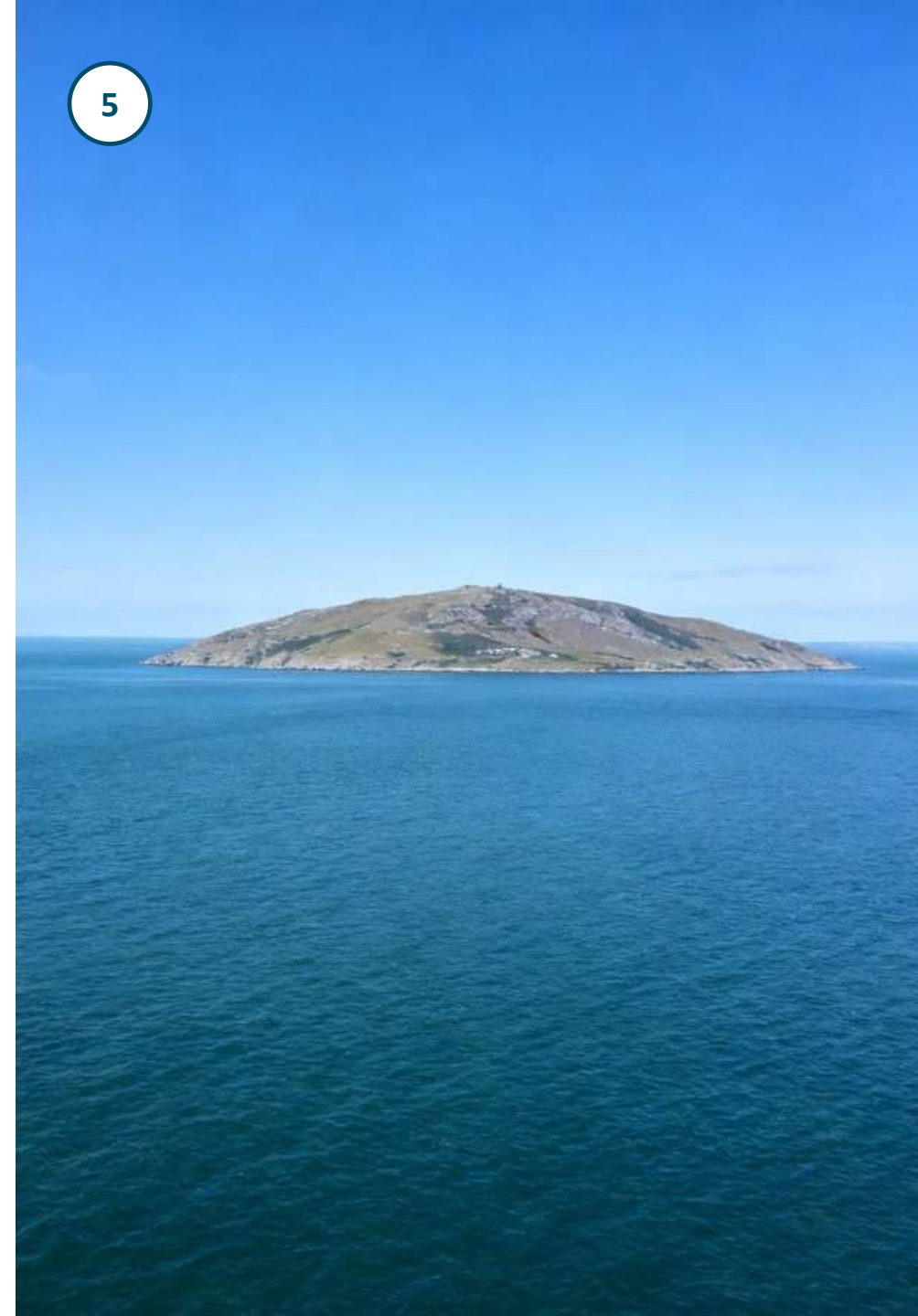
All repetitive work. Many scripted procedure runs and report executions, started by hand, one at a time.

Spanning a large number of teams. Storage, backup, datacenter work, configuration records, all still at zero automation.

Our team consists only of three people. Two others are not permanent and sit in other departments most of the time. Building automation takes time.

Plenty of opportunities. Not enough capacity to reach them.

fluvius.



No two islands worked similar

Some automated. Some did not.



Some did it all by hand.

Others handed it to a partner. Neither was a preference, both were simply what was at hand.



Where there was automation, it was whatever the author knew.

PowerShell, Python, Jenkins, Ansible, ... each on its own server, each with its own conventions.



Nothing shared, nothing reviewed.

When the author moved on it became a black box with a “do not touch” label attached, on a platform with no vendor support and an end-of-life date.

**We didn't have an automation strategy.
We had an automation archipelago.**

What it actually took us

4 · What we actually did



Three things — in the order we needed them.



Collaboration

You cannot convert a team that feels no pain.

You can let them watch one that did and thrives now.



Standardization

First, a team needs its own way of working. Then it can be standardized.

Only then automated.



Automation

One platform, with building blocks ready to use, so a joining team has little to learn and moves quickly.

Only the third one is an actual product.

Collaboration

Most of our customers are the business, not developers.



They don't want to consume an API.

They want the service, the same way every time. So they get a service catalogue, with the checks already built in.



We provided teams the building blocks on a standard platform.

Documentation

Guidelines

Onboarding

Development containers

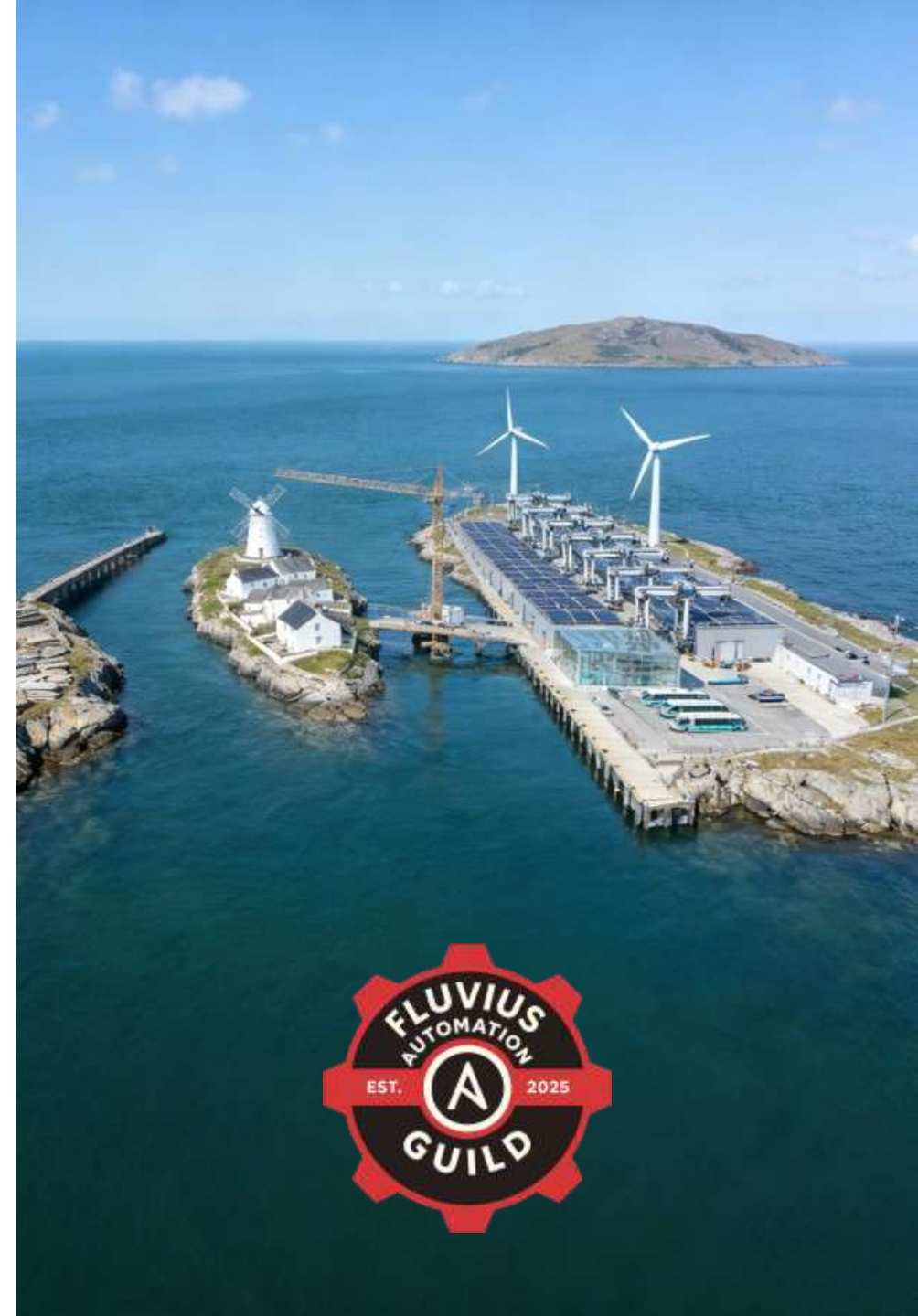
Our internal collections



The Automation Guild is how people find out.

About fifteen people, quarterly, at their location. Teams get curious before anybody asks them to use anything.

**Teams joined because it made their work easy,
not because they were forced to.**



The beginning of the journey



Until recently we were part of infrastructure. We now stand on our own.



We started with our own island first.

Infrastructure automation, for our service partner.

Every building block in this platform, originated from their needs.



We only recently became a team of our own.

We can now reach across the business instead of inside infrastructure, which means teams can now see that we exist.



Two more teams have recently joined.

The network team and one of the application teams, alongside those already running on the platform by themselves.

Most of those bridges have yet to be built.

Standardization challenges



Undefined processes cannot be automated.



We start from their procedures, not ours.

Every team already has a way of working. We adapt it to automation, naming conventions, standardize it.

The same steps every time, instead of replacing it.

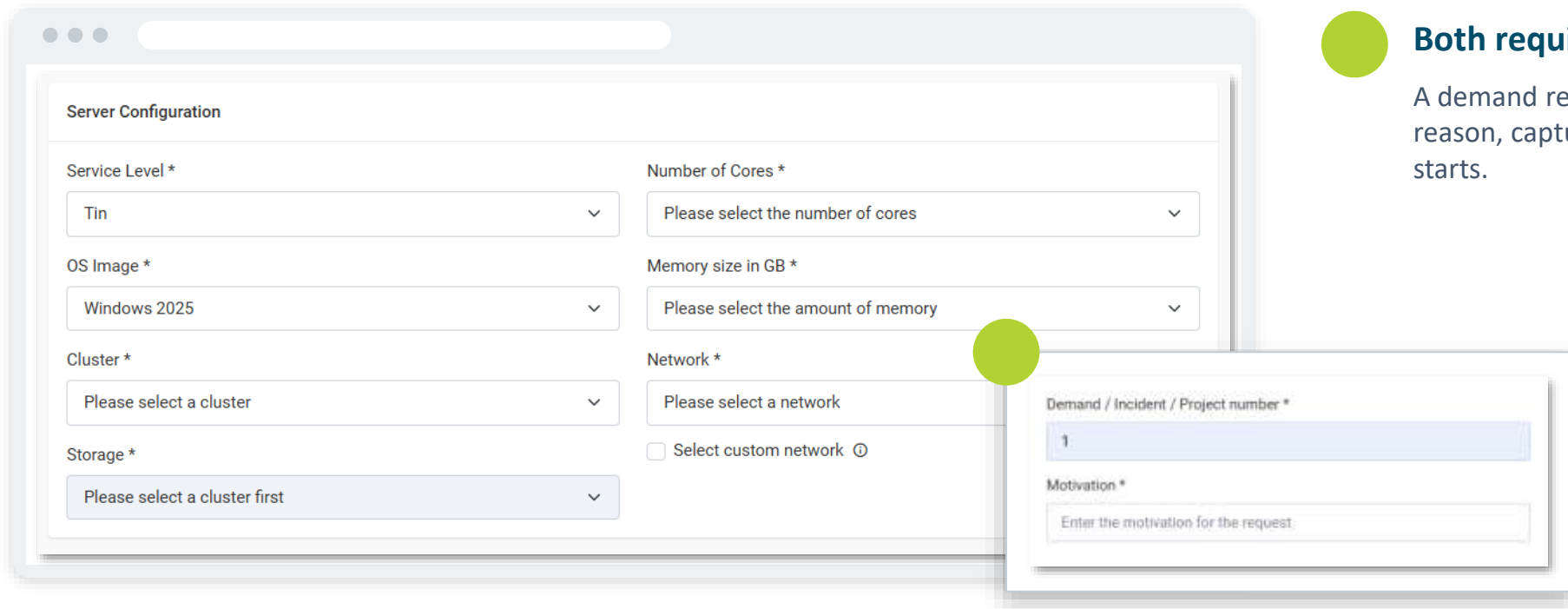
Our job is to take things away.

The spreadsheets go. The complexity goes. The errors are eliminated.

This is not the exciting part. It is the most important part.

Validation instead of spreadsheets

A form can only exist after the standardization. We build a digital service catalog.



The screenshot shows a web browser window displaying a form titled "Server Configuration". The form is divided into several sections with dropdown menus and checkboxes. A green circle highlights a modal dialog box that appears over the form, containing fields for "Demand / Incident / Project number *" and "Motivation *".

Server Configuration

Service Level *
Tin

Number of Cores *
Please select the number of cores

OS Image *
Windows 2025

Memory size in GB *
Please select the amount of memory

Cluster *
Please select a cluster

Network *
Please select a network

Storage *
Please select a cluster first

☐ Select custom network ⓘ

Modal Dialog:

Demand / Incident / Project number *
1

Motivation *
Enter the motivation for the request

Both required.

A demand reference and a written reason, captured before the work starts.

The form now does the validation. The automation just does the work.

The first win wasn't doing
the work faster.

It was catching any mistake
the moment somebody
made the request.

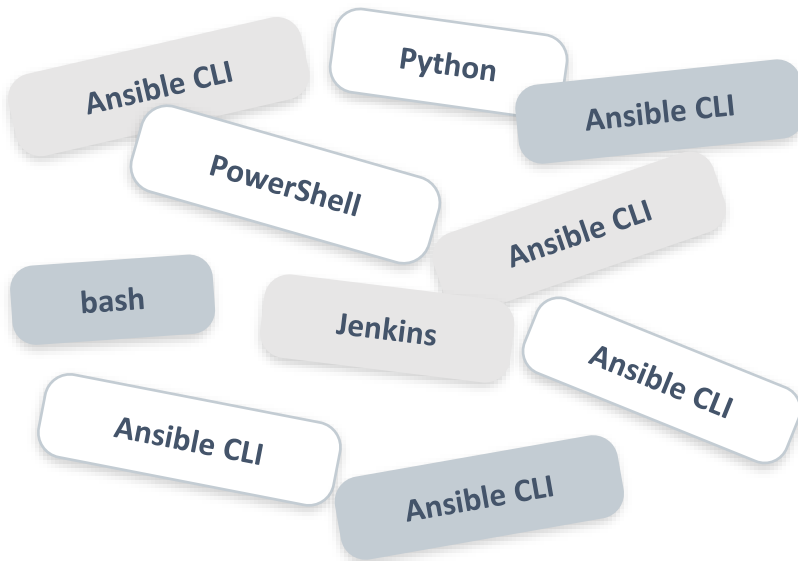


The automation platform



This is where the tool finally comes in.

Before



Now

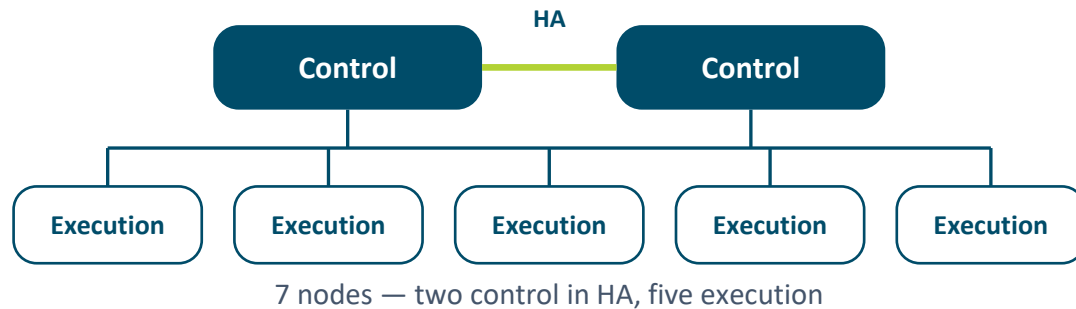


Creating the platform was the easy part. Filling it was the real challenge.

What we actually run



Ansible Automation Platform 2.6



Red Hat Enterprise Linux

13

execution environments

Some teams have their own: different firewall rules.

What runs on it

300+

job templates,
plus workflows

60

projects,
all syncing green

10.000+

hosts across
130+ inventories

90+

teams in
15 organisations

No OpenShift yet — this runs as containers on Red Hat Enterprise Linux.

The Fluvius collections



Simplified collections for routine work, so moving to Ansible is easy.

What the business uses

Routine tasks as roles, with a minimum of inputs:



Take a VM snapshot



Manage change tickets



Manage a maintenance window



What the collection takes care of

(So the teams do not have to)

Guardrails

A role exposes the routine task, never the whole system.

Normalization

Every input brought into one standard form, every time.

Complexity

The integrations and the order of the steps stay inside the role.

Management

Handled once, in the collection, instead of again by every team.

The business enables its own routine tasks, in a very short time.

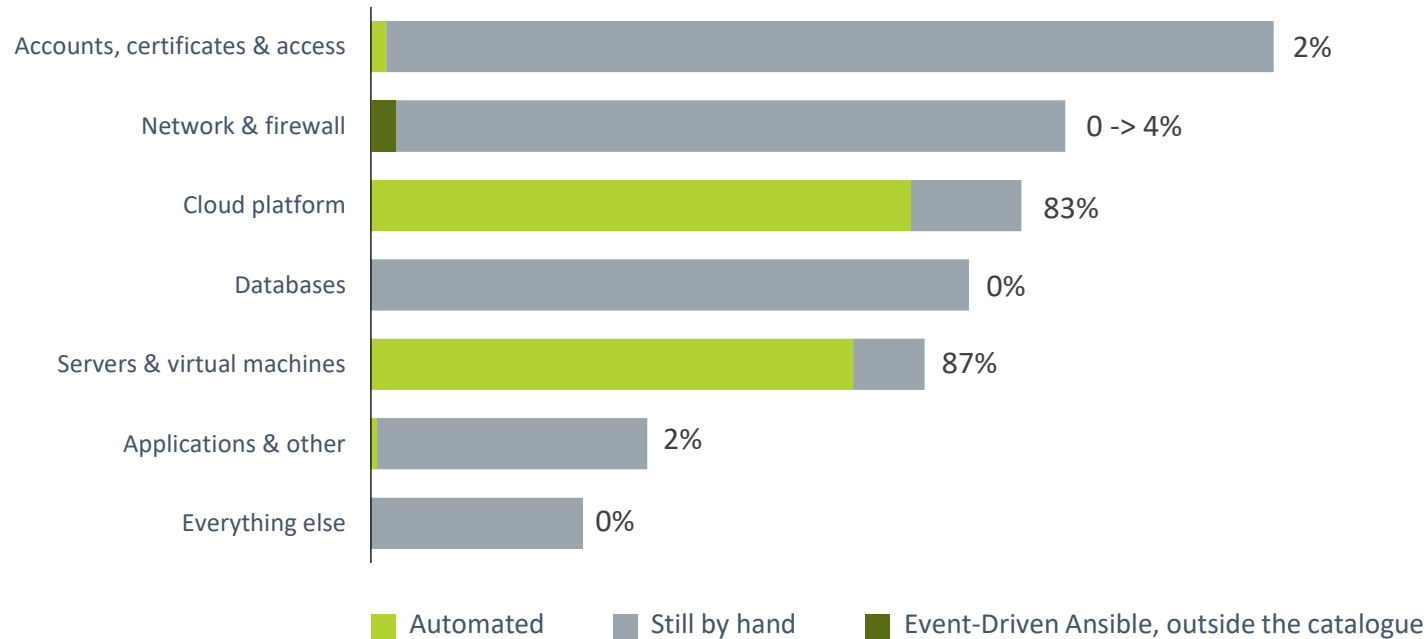
Where the automation actually is

5 · What changed



Before, we showed you 14.000+ requests done by hand.

There were actually 20.000+ of which 5.400+ already automated.



Also automated, outside the catalogue

A few processes are not requests at all. We moved them to Event-Driven Ansible, so they never appear in these numbers.

2 categories out of 9. Everything else is still where it was.

What infrastructure coverage looks like



The infrastructure catalogue - our own island, working for our service partner.

87%

of the server and virtual-machine catalogue is now automated.

22 infrastructure request types out of 29.



We covered the whole routine lifecycle.

Create, decommission, restart, snapshot, resize memory, CPU and disk, permissions, monitoring, software deployments.



What is left is short and specific.

370+ requests still by hand, and 290+ of those are a single type: Extending a virtual disk on a Linux VM.

Those are 2.500+ requests nobody had to pick up, retype, chase or wait for.

Five days became two hours

5 days

to deliver a server — and what arrived was a machine and a to-do list.



2 hours

for the same server.
Registered everywhere and ready to use.

Created August 12, 2026 at 3:20 PM
Updated August 12, 2026 at 5:35 PM
Resolved August 12, 2026 at 5:35 PM

One real request, raised to resolved.

The work was not the problem. Waiting for the other islands was.

Automation at scale



Not all jobs originate from the service catalogue. Many support the operation of our business.

27.000+

automation jobs since
October 2025

+137%

increase of running jobs from
start and still climbing

(~1.600 → ~3.800 jobs a month)

52%

run unattended — scheduled
or as a part of a workflow

1.500+

different hosts
automated

More than half of all automation now runs unattended.

Unattended: 52%

Started by a person: 48%



Share of all launches, last 120 days of job history

We onboard the teams. After that they are autonomous. They decide their own work and build their own automations. We supply the guides, the collections and the advice.



Use case: a self-managed Privileged Access solution



The application owner manages solutions with the CMDB. Ansible does the work.

No forms

No approvals

No request

End-to-end automated



Solution owner

Manages the member servers and permissions in the CMDB.



CMDB

The single source of truth for every solution and who may manage it.



Ansible Automation Platform

Compares the CMDB with PAM and applies only the difference. Our own collections, 67 modules, API to API.



PAM

Only the authorized administrators get access to their solution.

Eighteen minutes to prove nothing had to change.

One example run, 23 September

1.000+

servers compared

2.000+

PAM records verified

17.5 min

median run time
200+ runs in 120 days

0

changes needed in that run

Use case: a self-managed Privileged Access solution



Lessons learned - nobody is perfect



Parallel writes caused database deadlocks.

Thousands of deadlocks a day, measured by the database team. Every write loop is strictly serial now.



We added a brake.

If one run would revoke access on too many servers at once, it stops instead. A bad dataset should not quietly strip the full PAM tool.



Never guess at missing data.

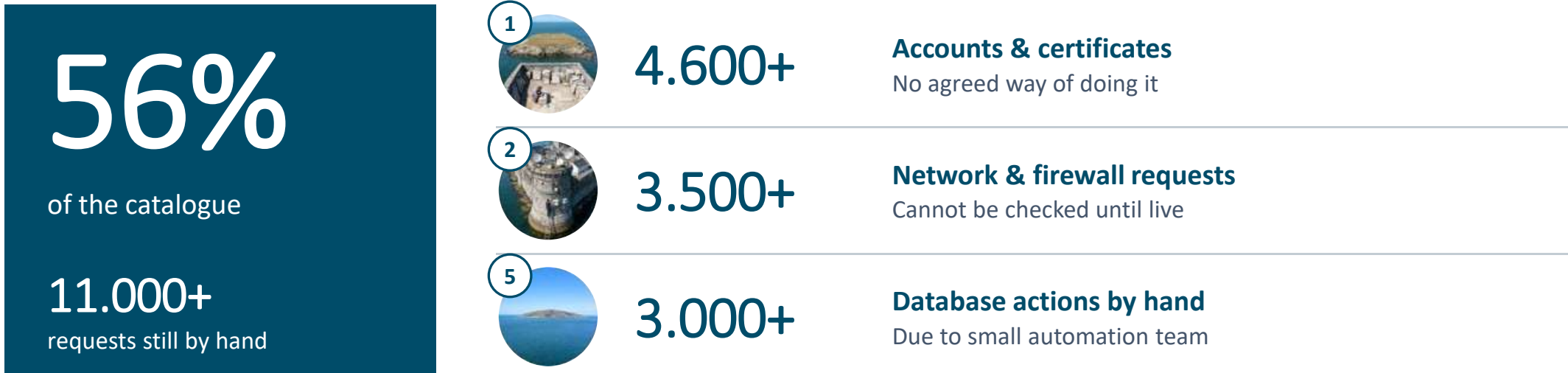
Text in various formats should not mark a server as invalid. A temporary lookup failure changes nothing at all. So the result always matches the standard we defined.

Every one of them made the automation slower, on purpose.

What we have not touched



Three categories, between zero and 1.7% automated.



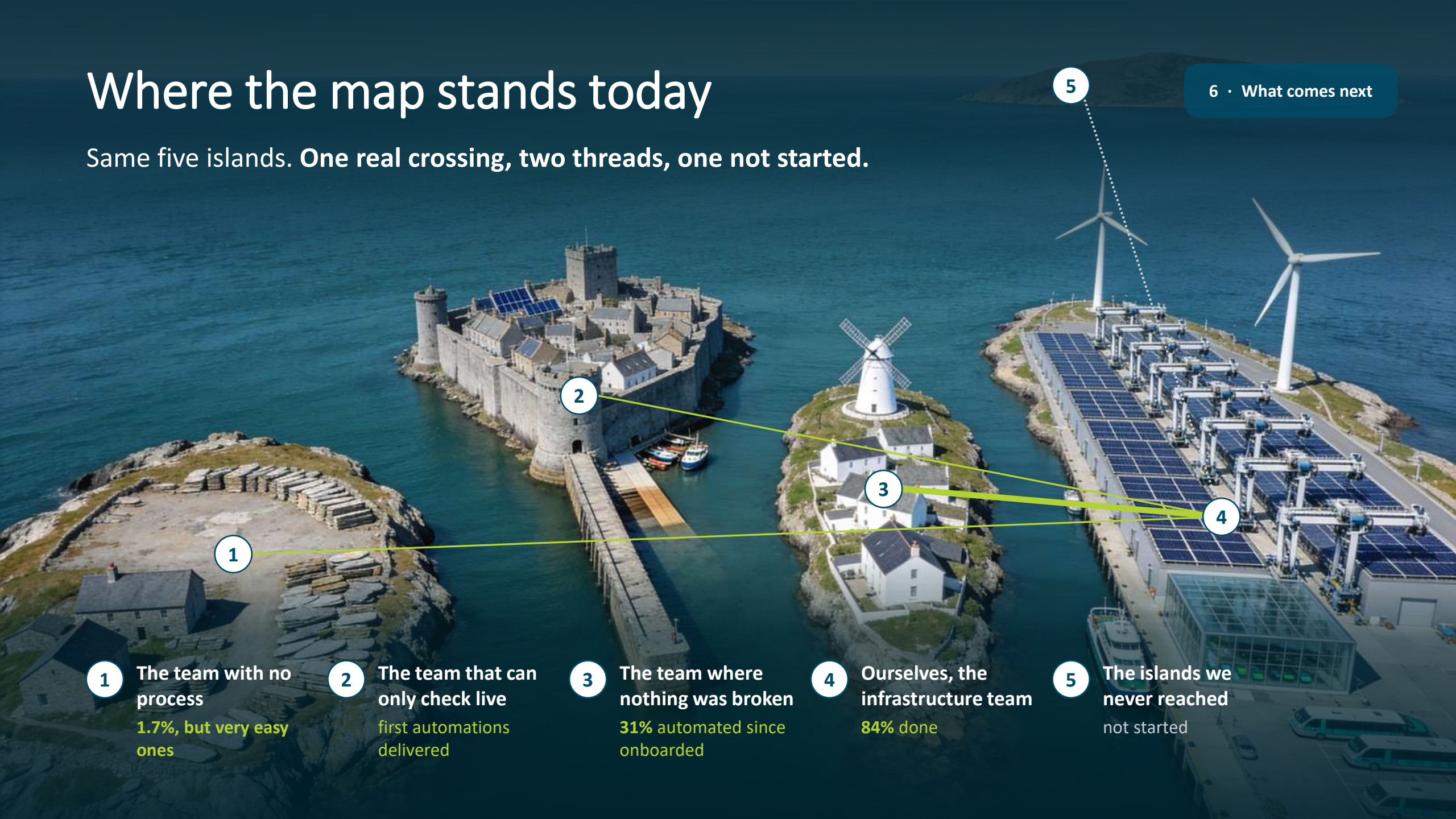
The biggest one is also the most concentrated. Four request types are 90% of accounts and certificates work. It is the largest category in the catalogue and the next one to pick up.

Three different reasons. Being too hard is not one of them.

Where the map stands today

Same five islands. One real crossing, two threads, one not started.

6 · What comes next



1

The team with no process
1.7%, but very easy ones

2

The team that can only check live
first automations delivered

3

The team where nothing was broken
31% automated since onboarded

4

Ourselves, the infrastructure team
84% done

5

The islands we never reached
not started

Next steps in our journey



Every team on the platform for routine work. Not for the hard work.



The biggest category

Accounts and certificates: 4.600+ requests of which 4 request types count for 90% in total.

The island with no process



Activate the teams on board

Six of our fifteen organisations have projects, permissions and people.

None of them has ever run an automation job.

Team adoption



Not the expert work

The mandate covers routine requests only. Tasks requiring expertise remain with the specialists.

Keep the scope manageable

It is no longer about the tool, but about the processes.

If you take 4 things home

1

Go where you are welcome

Not where the volume is. A team that wants to automate makes it happen in weeks. A team that does not automate will argue for months.

2

You cannot automate an undefined process.

Automating an undefined process only makes a faster mess. Defining the process took longer than automating it. Every time.

3

Be realistic. Ignore the fairy tales.

A talk with only good numbers in it is a brochure. The real numbers may be confronting at first, but at least you can measure progress.

4

Keep human factors in mind.

Show people how technology can help without taking away control. Acknowledge their expertise and show real use cases to inspire.



None of the 4 is about a tool.

3 of them are about people.

Build bridges, not islands.



Red Hat
Summit

Connect

Thank you

Arthur Odekerken

Expert Systemen en Operaties

Jurgen Van de Perre

DevOps Engineer

> Next session in this room @ 14:00:

Introduction to Ansible Automation Orchestrator

Live demo by Red Hat

