OpenShift @ TGW

Full-stack Deployment von Hardware, OpenShift, TGW Applikationen und kundenspezifischen Anpassungen in Kunden-Infrastrukturen

TGW LOGISTICS

What (not) to expect

- This is not a „best practices“
  - try to describe the environment we‘re working in,
  - its challenges,
  - how we try to mitigate / solve them

- If any questions come up please just raise them
  - If small enough I‘ll try to answer inline,
    worst case we need to postpone to Q&A / discussion at the end

- I‘m very interested in any recommendations, remarks or ideas
  - We definitely left room for improvement

About me

Klaus Linzner

- Software Architect @ TGW
- Strong Background in .NET – working on base .NET libs for years, some C# talks at conferences
- Couple years ago heading more into build & delivery mechanisms
  - Lead migration to git and azure devops
- Delivery mechanisms and builds were extended to software delivery to site
  - Lead on internal project ADP „Advanced Delivery Platform“ – the project that eventually introduced OpenShift

WHAT WE'RE DOING

warehouses

millions of crates

hundreds of devices

hundreds of times

what we‘re doing

a single device can be…

a storage rack

what we‘re doing

a single device can be…

a shuttle block

what we‘re doing

a single device can be…

a (part of a) conveyor

what we're doing

a single device can be...

an AMR / AGV

what we're doing

a single device can be...

a palletizer

what we‘re doing

a single device can be…

a pick station

what we‘re doing

Depending on customer needs we deliver everything from ground up.

Each site is scaled / designed to customer requirements (number of transports, storage locations) which determines compute resources.

But of course: Standardization is a big factor

what we‘re doing

The workload we‘re running is usually rather static.

Peak times usually black friday to christmas.

(can go so far: last chance of deployment for the year is middle of october – next time middle of january)

what we're doing

In general, our software ~~should~~ needs to be able to run entirely air gapped.

Some customers require it (at least in theory).

Maybe more valid: They want to be able to run it without external dependencies.

what we‘re doing

Many (not all) of those components are time-critical!

With those time constraints, our software usually needs to run in the customers datacenter(s).

This means within the customers infrastructure.

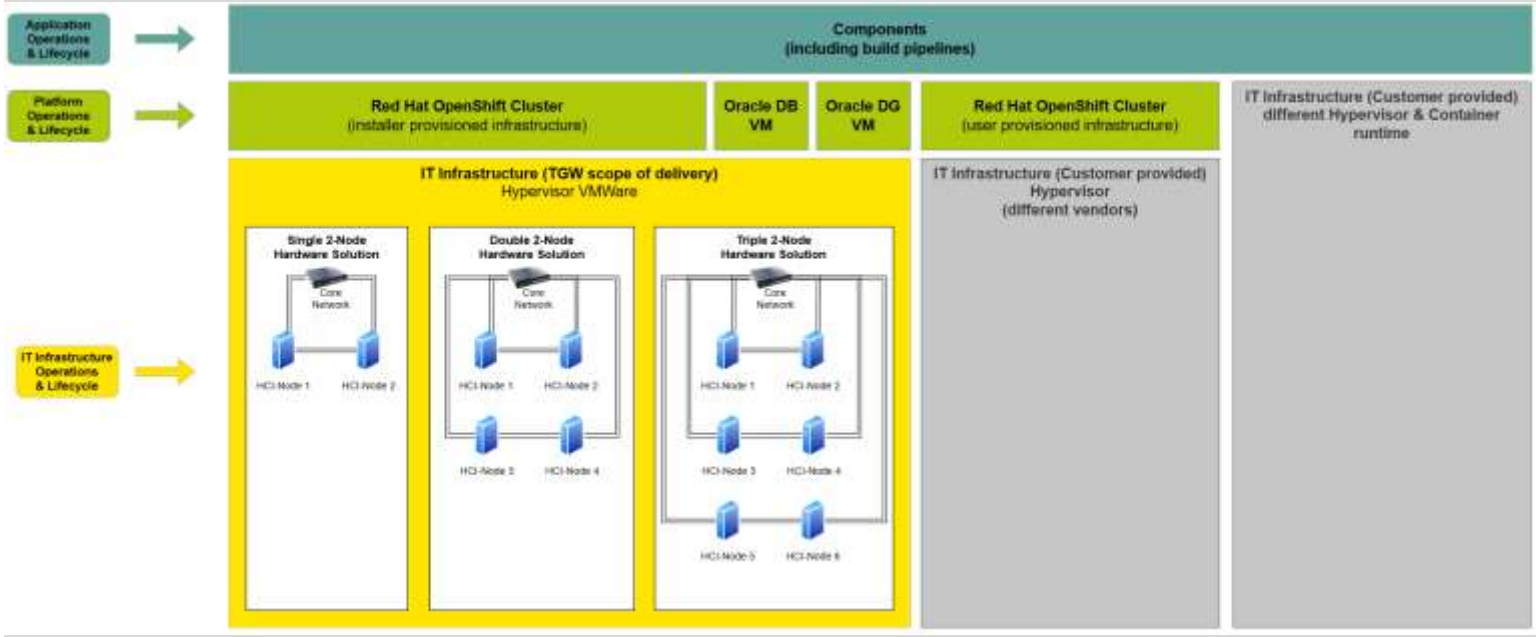
what we‘re doing

Two approaches to running in customers infrastructure:

Black box (preferred): We ship servers, racks,… As much as possible. And run everything.

Best case: We need a power source and a network cable*

10 years ago this was the default.



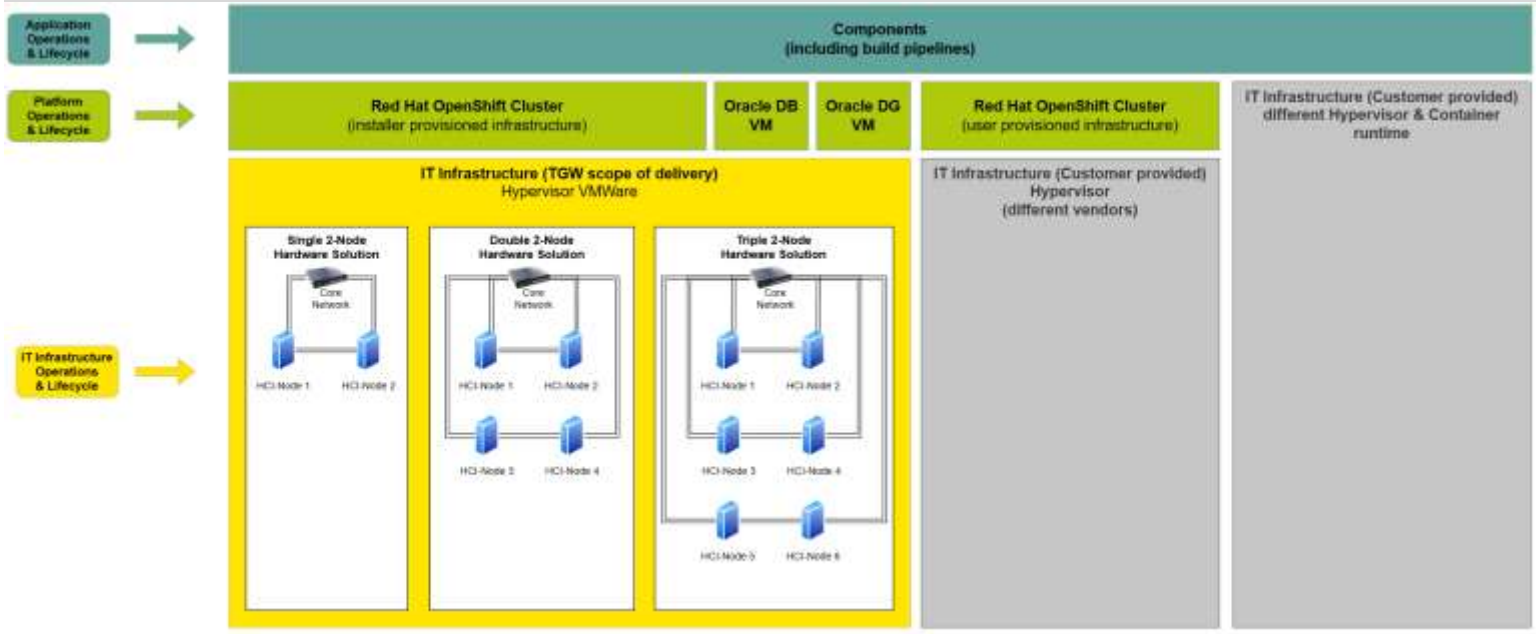
*of course it‘s not that simple as we‘re talking about several different VLANs

what we‘re doing

Grey box: The customer runs hardware, and we run on top of the hypervisor.

As remark: we never ever run directly on the hardware – always a hypervisor in between

(hardware patching,...)

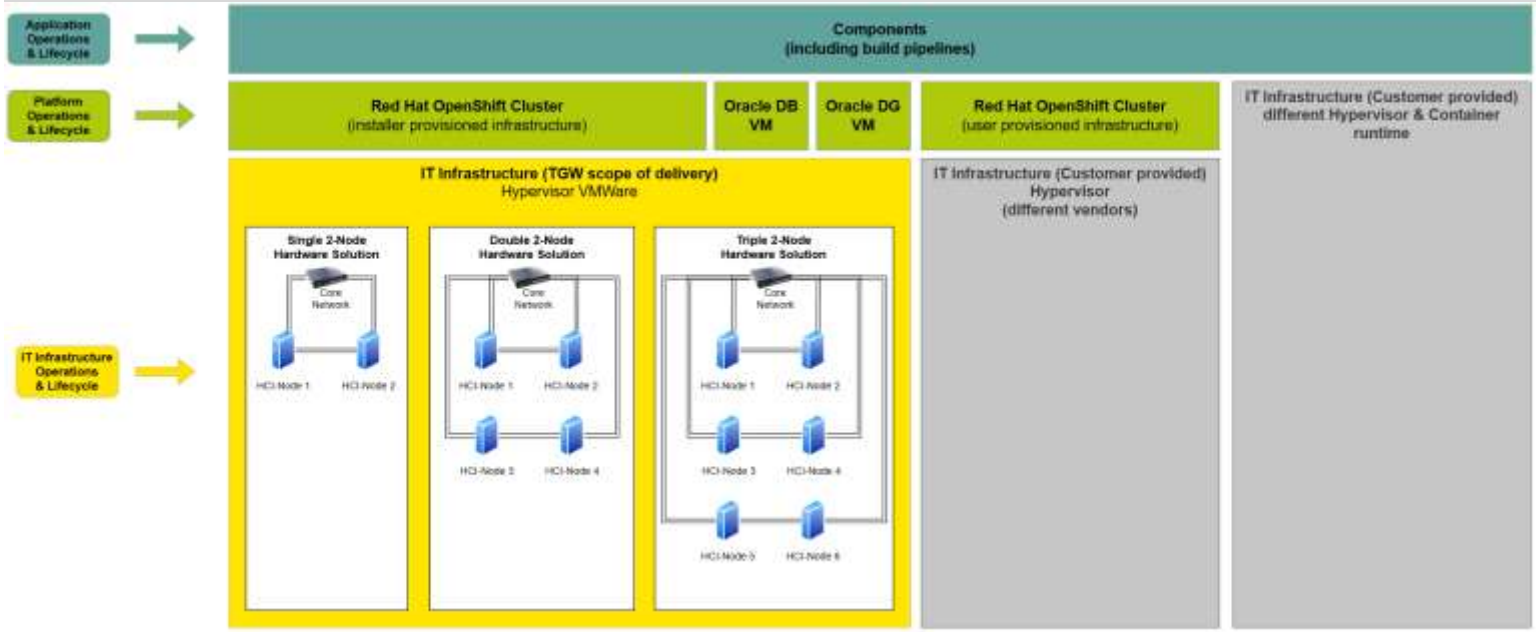


what we're doing

Various shades of grey however:

We ship virtual machine images and customer scans/imports and runs in their environment

Customer provides virtual machines, we install into,



how we came to openshift

Back in 2022

Obviously - all of those devices and their controlling software needs to work together smoothly.

What was NOT obvious for a long time – that all of the software should follow same patterns for deployment – making it easier to use and to update

Back in 2022

The basic software products are developed by different teams.

In different programming languages.

With different frameworks.

Different patterns.

Order of magnitude: 10 teams - 100 developers.
Mostly Austria & Germany

Back in 2022

Product teams ship software components to realization teams.

Some software needs only configuration.
Some software needs customization (customer specific code).

Realization teams integrate the different software parts.
Build it with different tools.
Deploy it with different mechanisms.

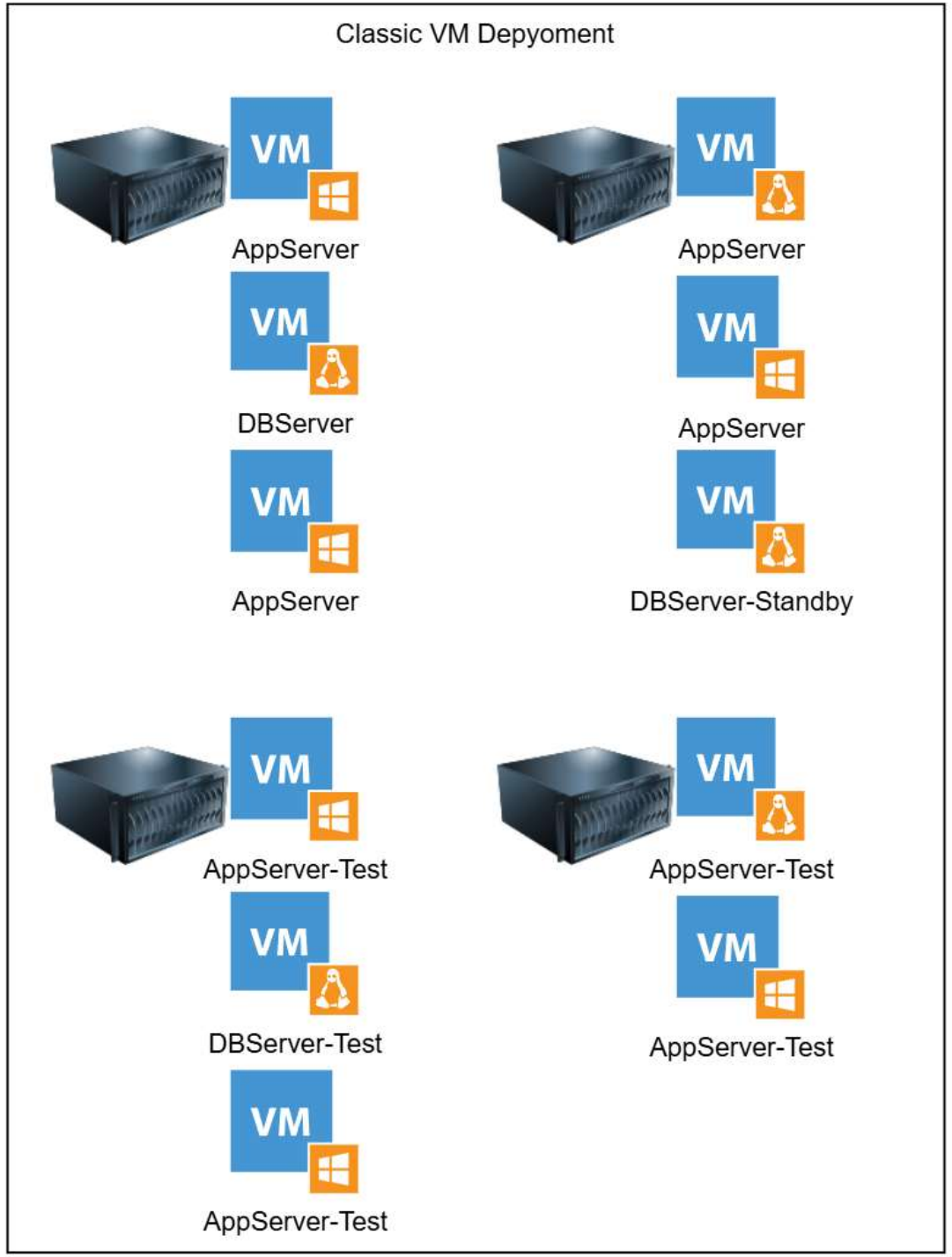
Order of magnitude: lots of changing Teams, 200 developers Worldwide

Back in 2022

Every time one of those 300 developers needed a new (in-house) test environment it meant the following:

- Provision 3-5 virtual machines (windows / linux)
- Install required components
- Install required cross-functional services (messaging / authentication)
- Configure all of those

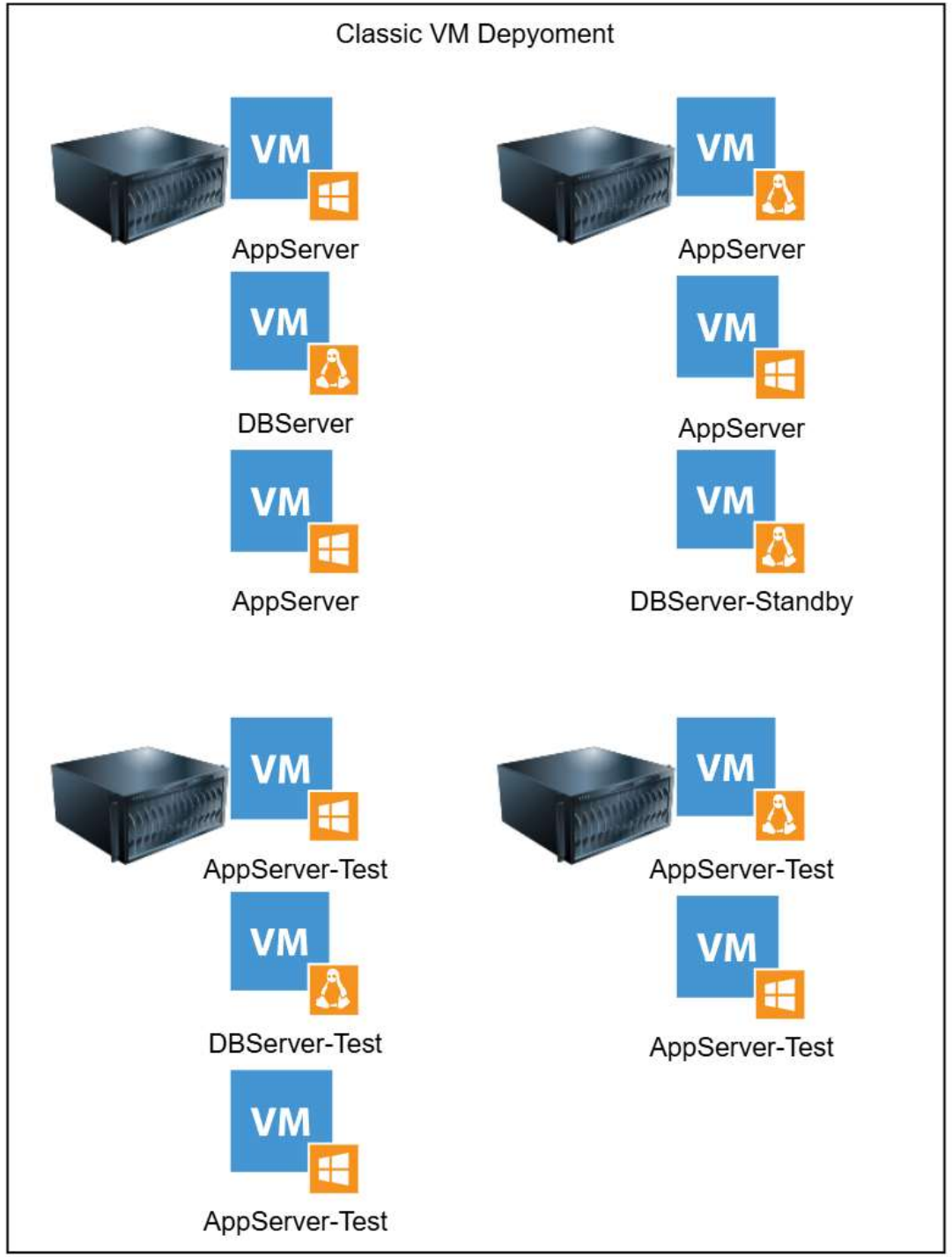
- In some cases: It took a week per test environment (OR: beg your teammates for their environment)



Back in 2022

Every time a developer needed to update…

- Run builds
- RDP/SSH to machine(s)
- Stop service(s)
- Run Installation (different tools)
- Apply (static) configuration
- Start services



Back in 2022

Roughly the same time: We already started consolidating to git / azure devops.

Some components already were there

Some had it on their roadmaps.

What was unclear:

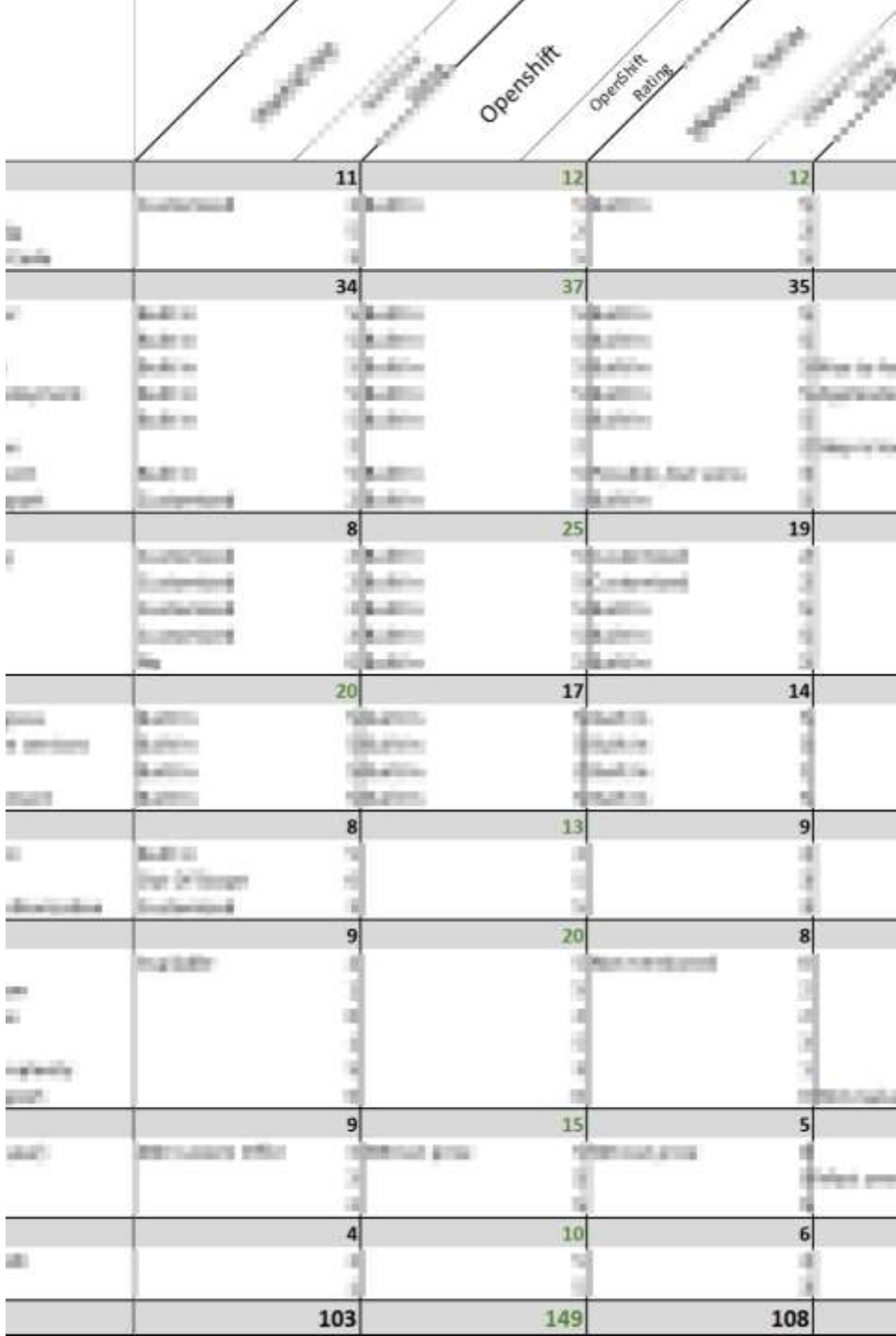
- How do we want to ship software to sites?
- How can we update / patch software / operating system?
- Which features do we need from the system?

Back in 2023/2024

Long story short – a year of evaluation and testing.
(In parallel: containerizing applications)

Despite initial reservations kubernetes / OpenShift came up top

- Patching / updating including operating system
- Training material
- Internal DNS / service communication without huge service mesh
- Certificate offloading
- OpenSource



Back in 2023/2024

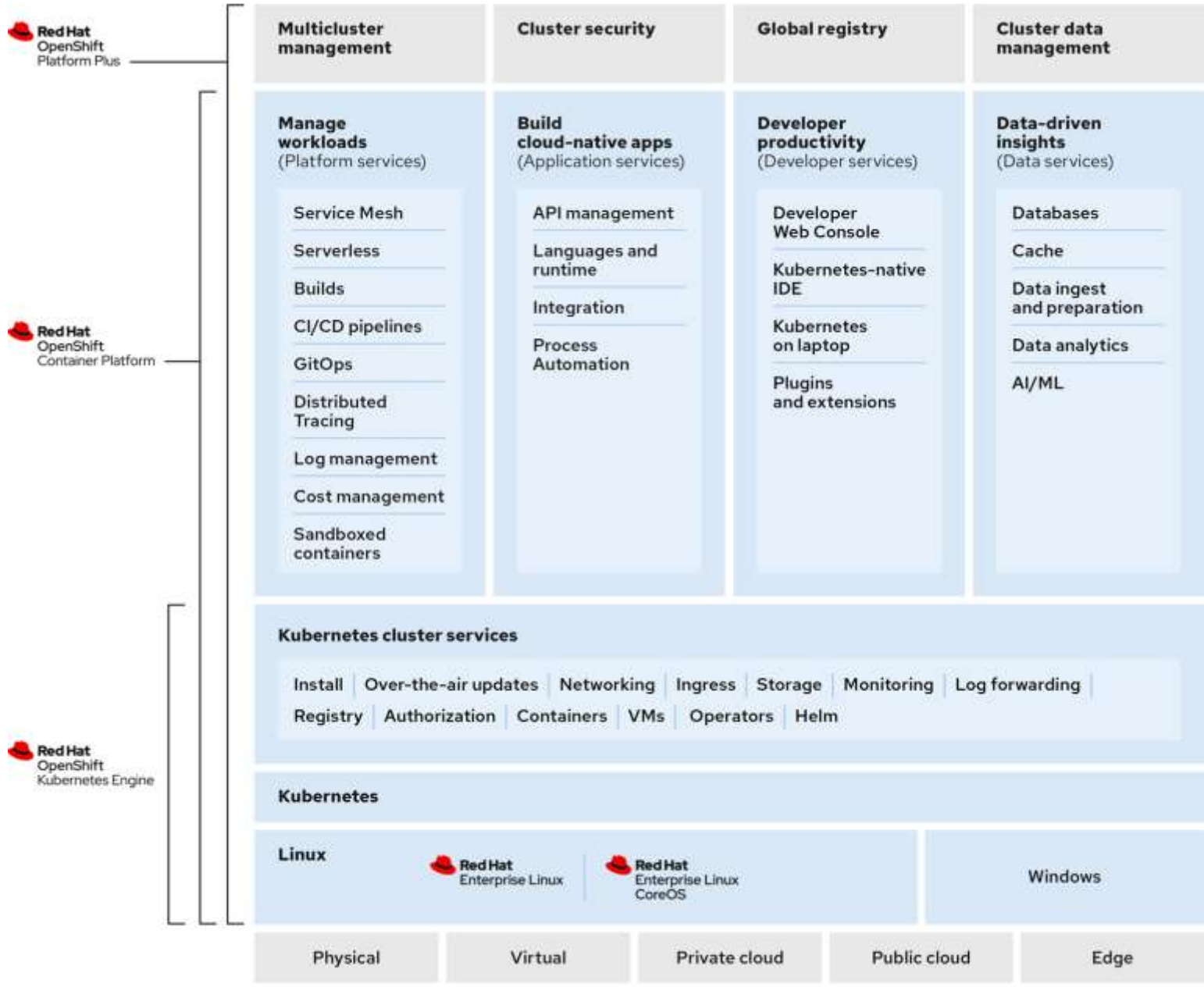
Two notable execeptions / remarks for migration:

- Oracle db remains as-is
- We‘re using OpenShift Kubernetes Engine

OKE has some downsides as we can‘t use features provided by OpenShift and that we have to implement ourselves

(we‘re delivering our own argocd, log aggregation, messaging,...)

But: direct pricing considerations.



INSTALLATION

@ CUSTOMER

Installation

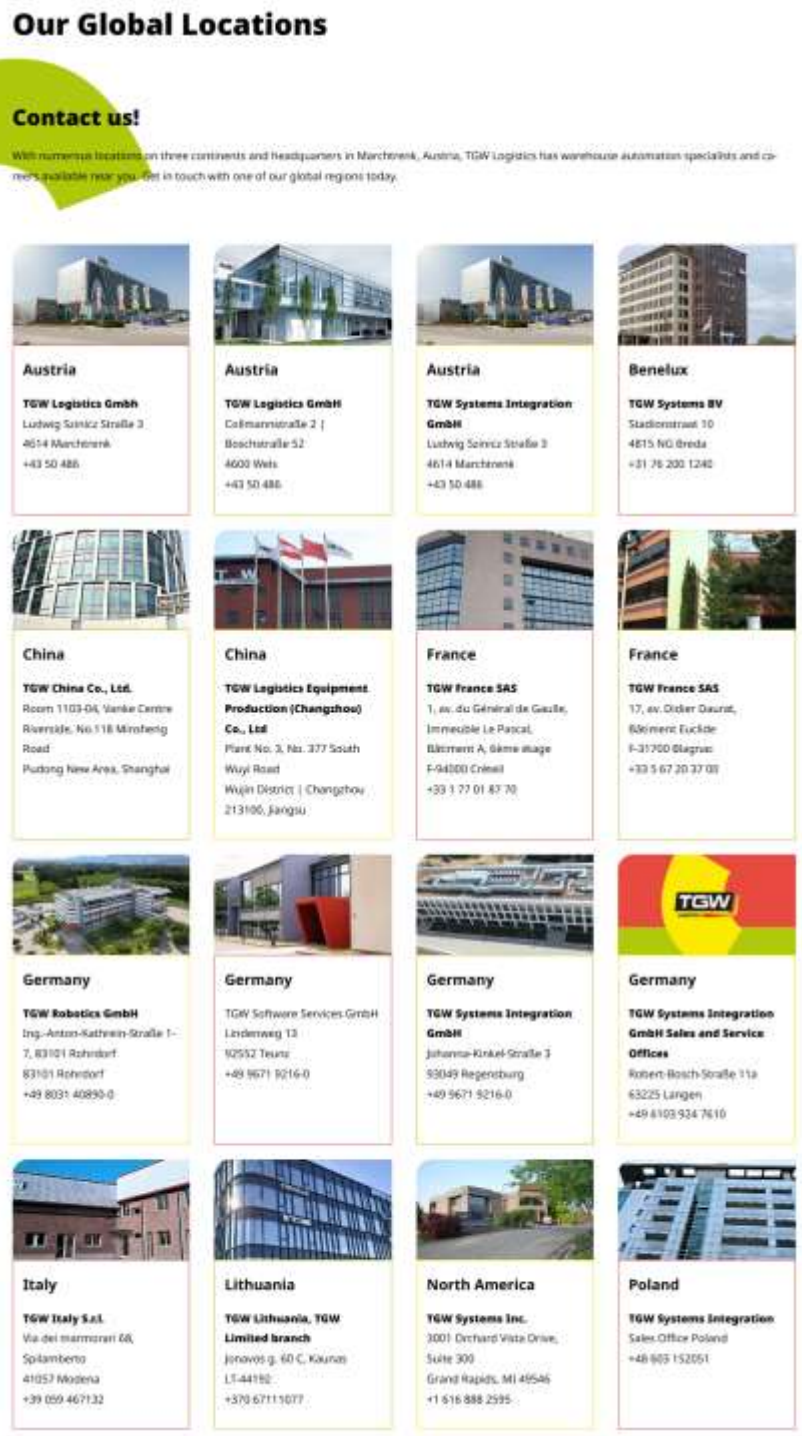
Up front: As of September 2025 we installed the new system with 7 customers.

North America committed and started late 2024 with two.

Northern Europe committed soon as well.

Central Europe was on standby and followed once Nothern America „was OK“

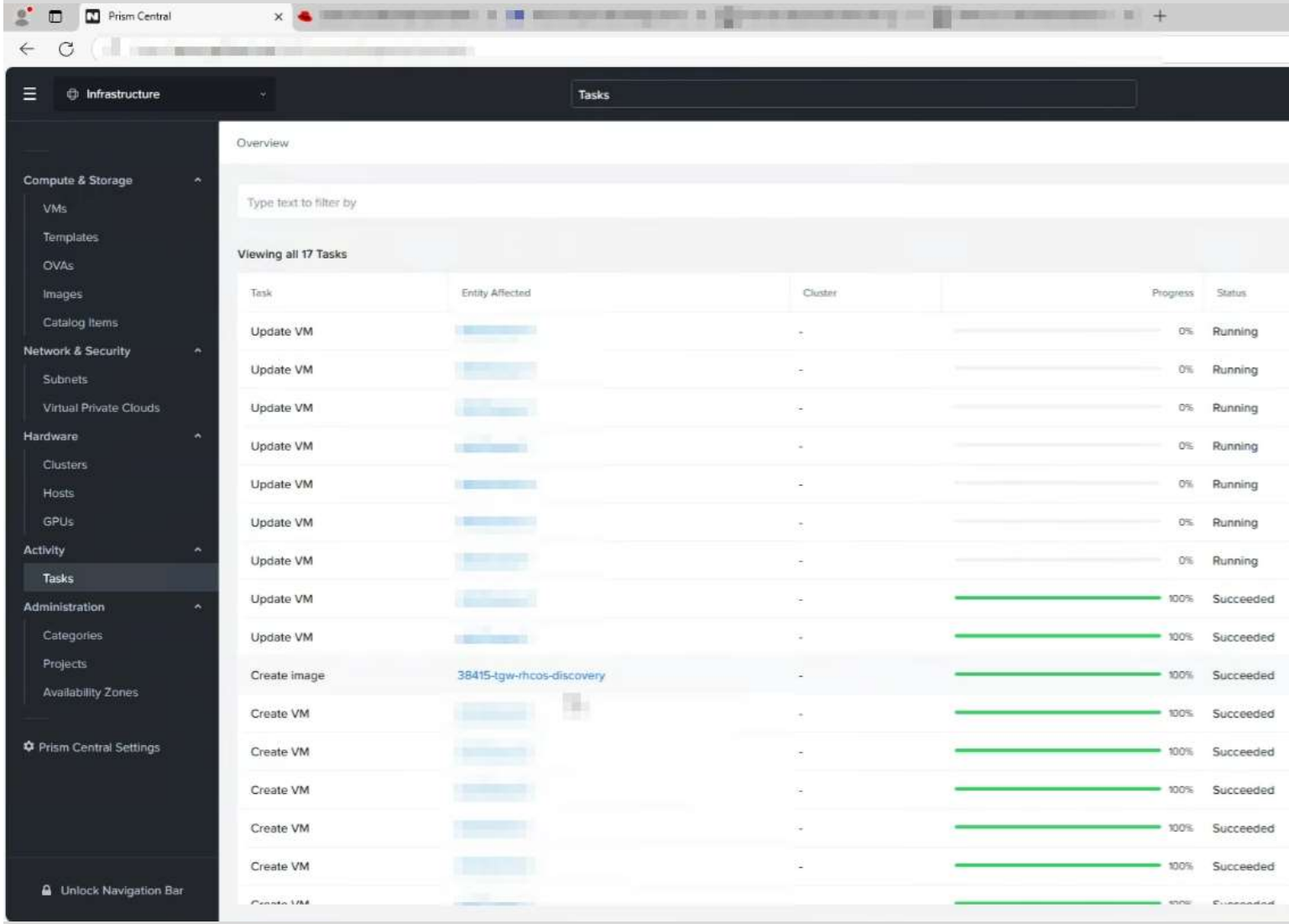
After initial concerns regarding our new Platform, teams wanted to migrate additional 15 – but we don‘t have capacity for it yet



Installation

Of those 7 customers

- All of them provided their own hardware
- 5 running VMWare
- 2 running nutanix
- 1 customer with vmware installation already requested reinstall on nutanix as they are migrating within a year
- 2 already running openshift, proposed to run in their cluster
  (we declined, not there yet)

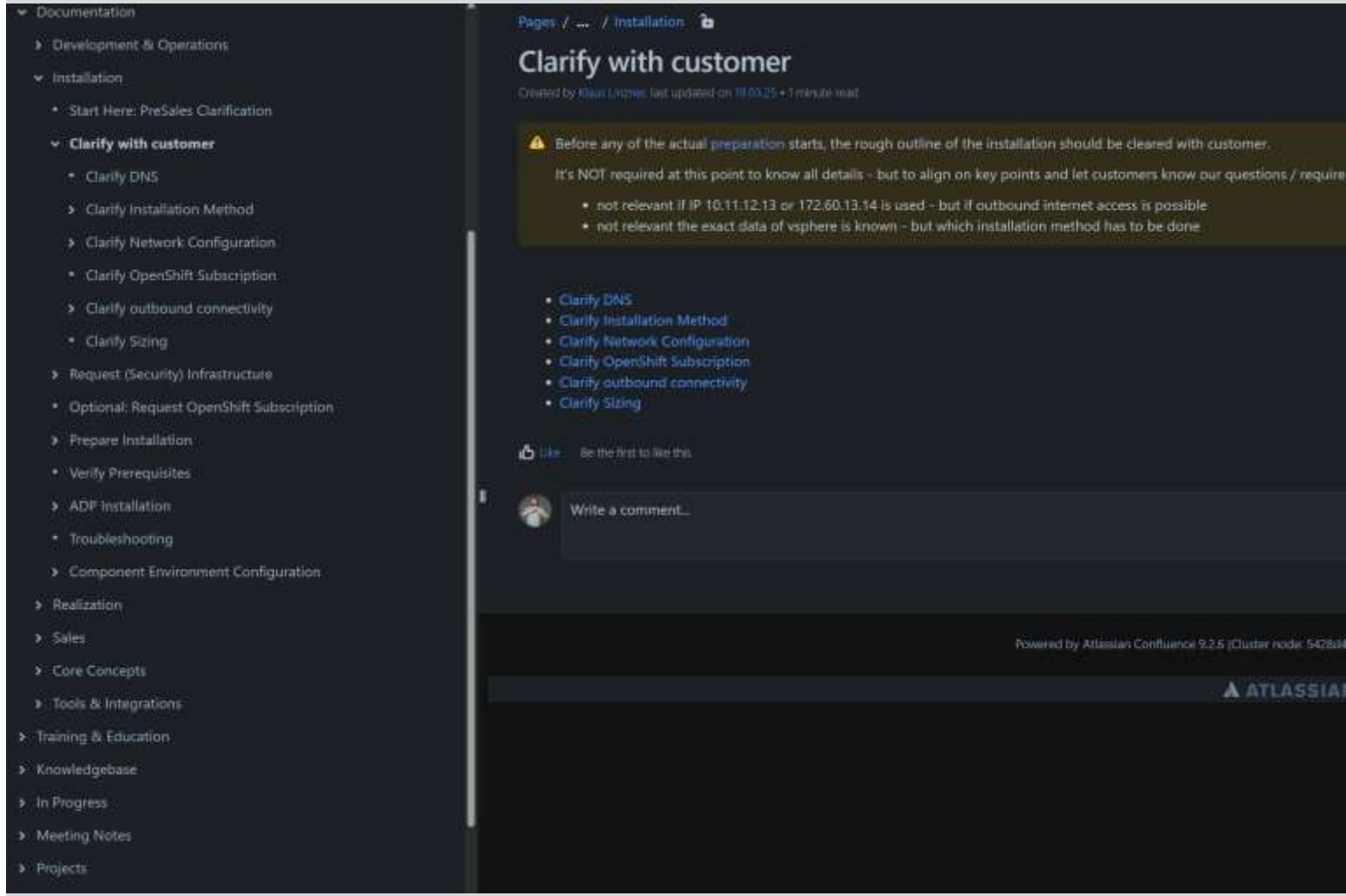


Installation

We‘ve created lots of documentation, cheat sheets,… for realization departments to clear with customer.

Some of the clarifications ~~should~~ need to be made BEFORE contracts are signed.

Some of the clarifications are made close to installation (sometimes a few weeks before installation)



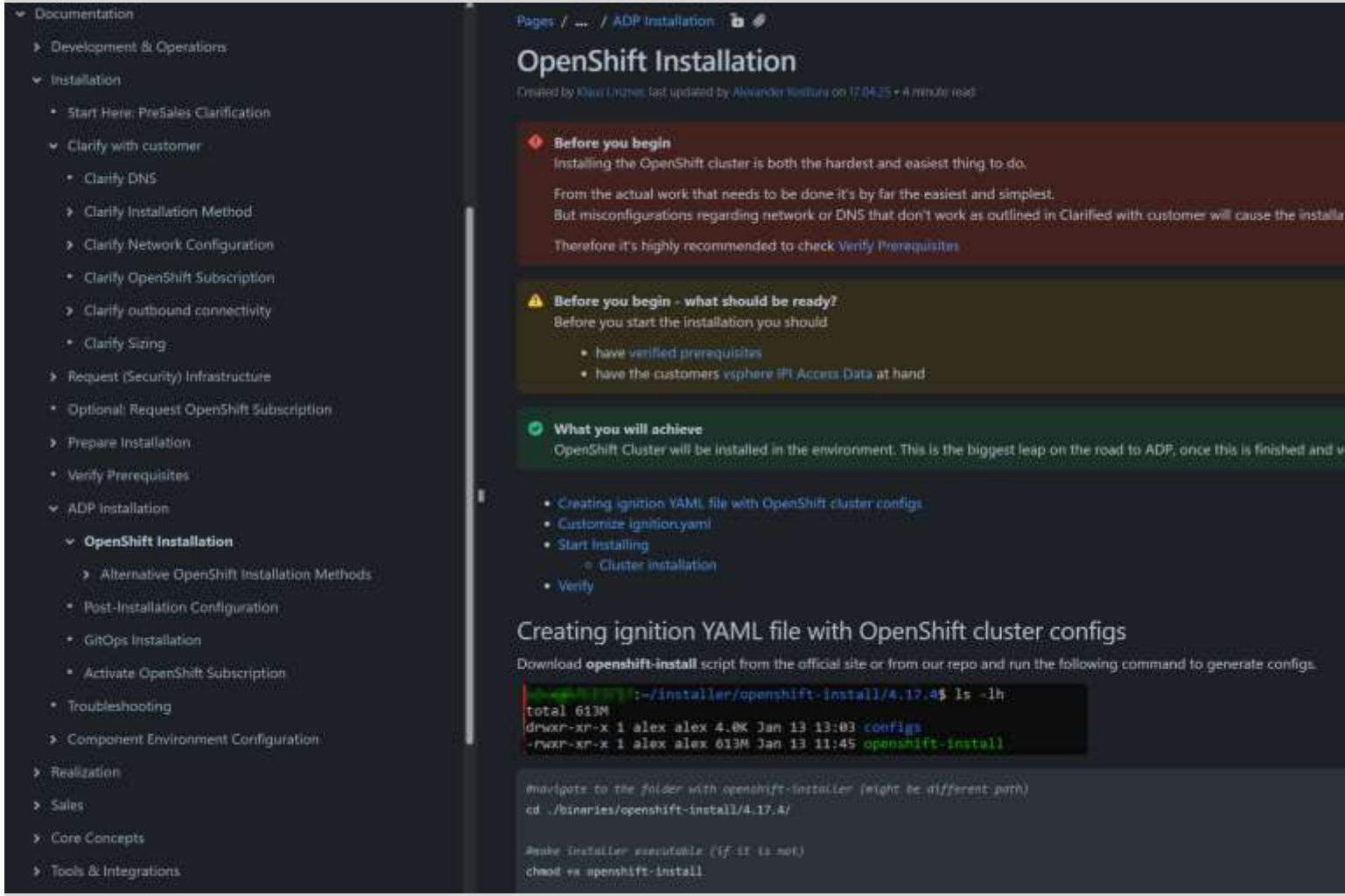
Installation

There‘s (rather) clear documentation in internal confluence.

Each of those customers we gave the detailed requirements up front:

- Required network connectivity
- Required hypervisor permissions

Each of those customers acknowledged and said it was done.



Installation

For the actual OpenShift Installation we‘re recommending openshift-install

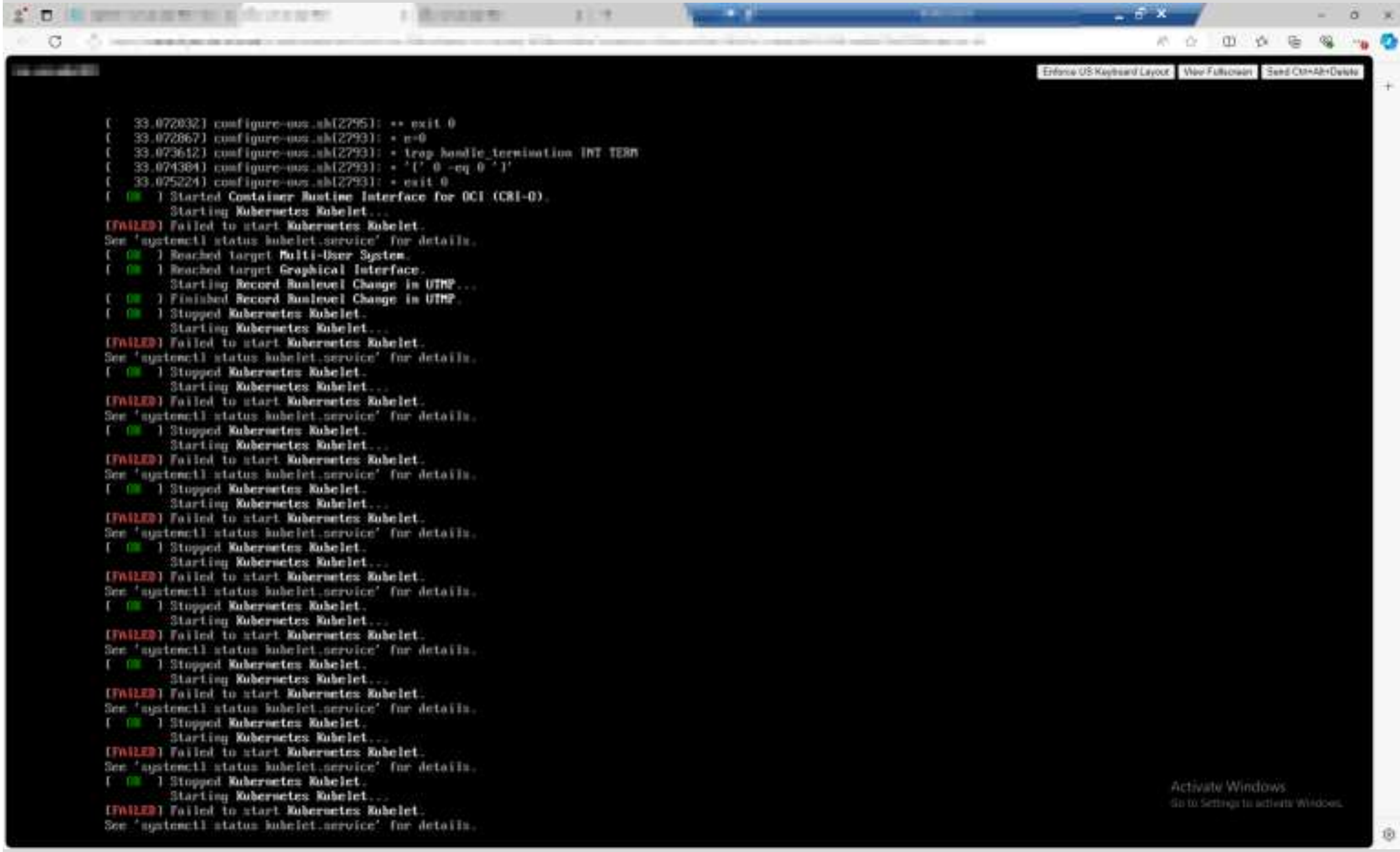
This works brilliant in-house.

Installation

This only worked for 1 customer.

Reasons:

- Network connectivity
- Hypervisor permissions
- Hypervisor connectivity
- Hypervisor internal screwups

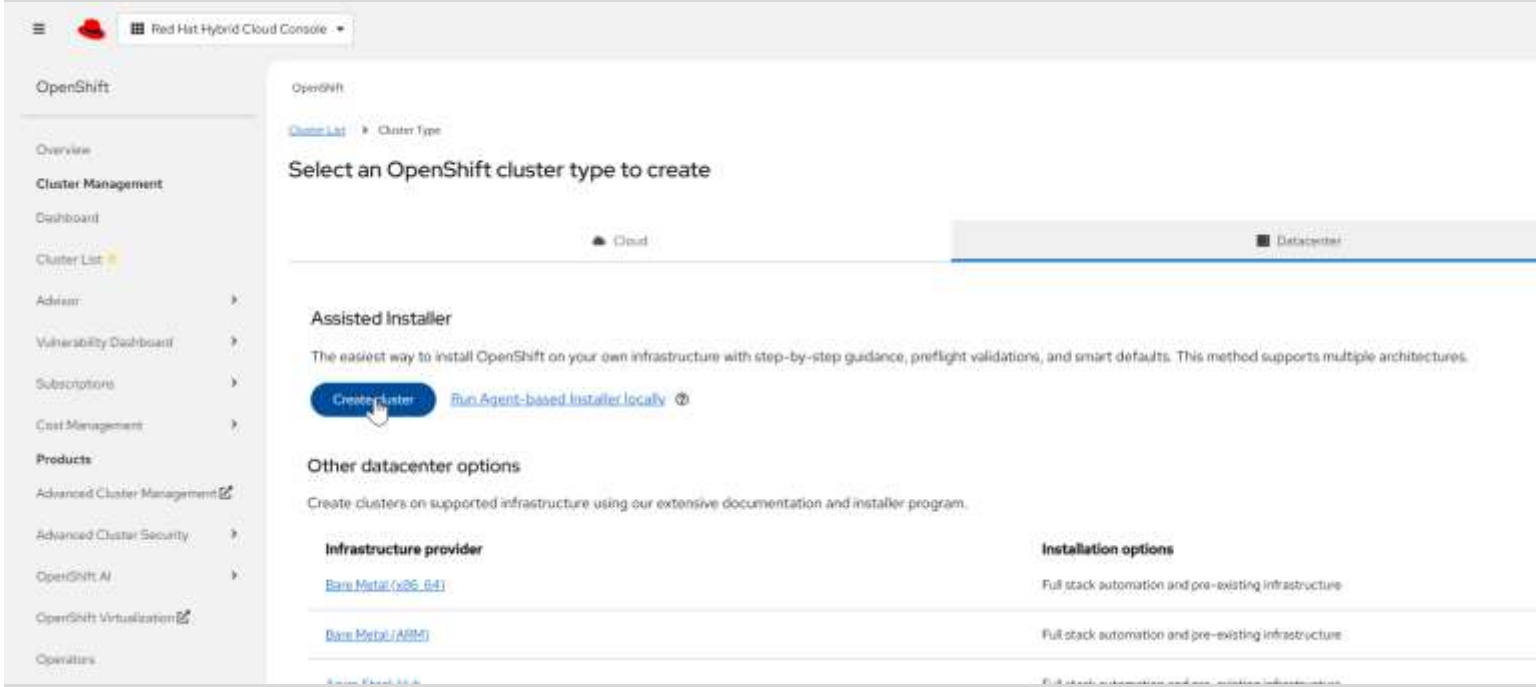


Installation

Fallback to UPI / assisted installer in other cases

Assisted Installer elimates hypervisor dependencies during installation.

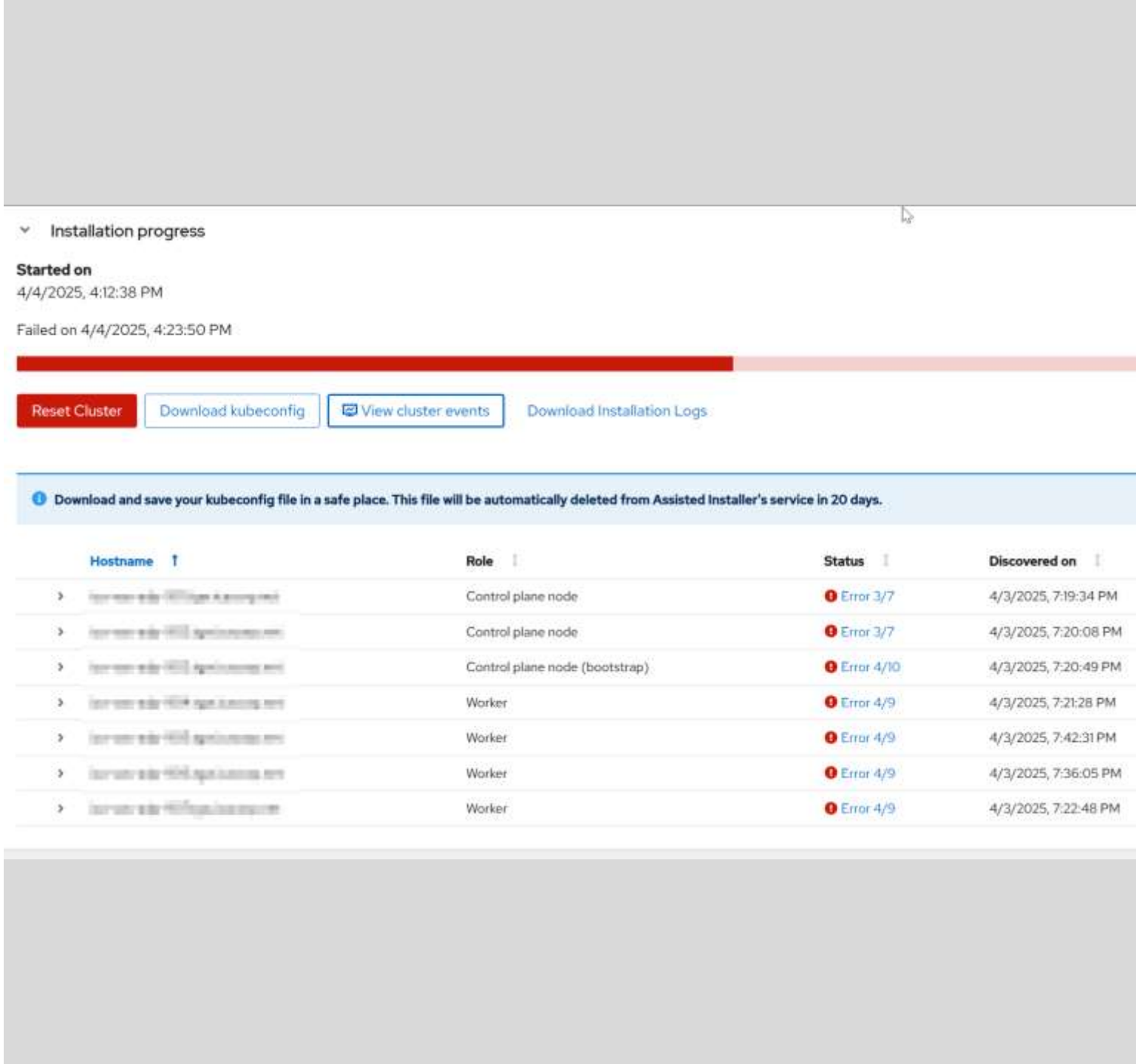
Much more stable / fault tolerant in terms of connectivity (timeouts,...) and more tolerant to resume when errors occur.



Installation

But – even there we had network surprises

(that assisted installer allowed us to work around - MTU default configuration in combination with quay and podman causes installation timeouts)



Installation

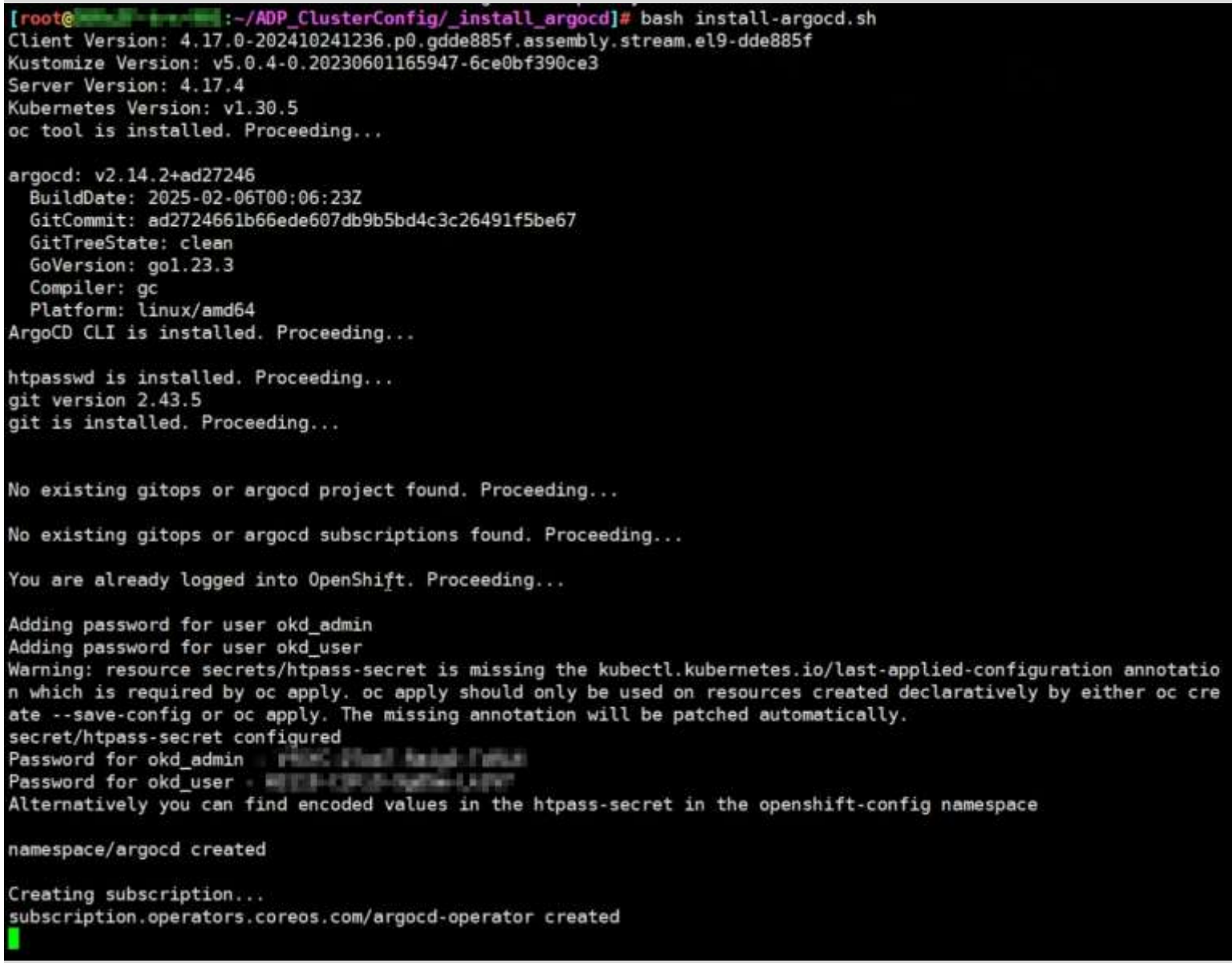
While the product recommendation is openshift-install,

the project teams are now heading to assisted installer exclusively.

Easier to install, less code required.

In all cases:

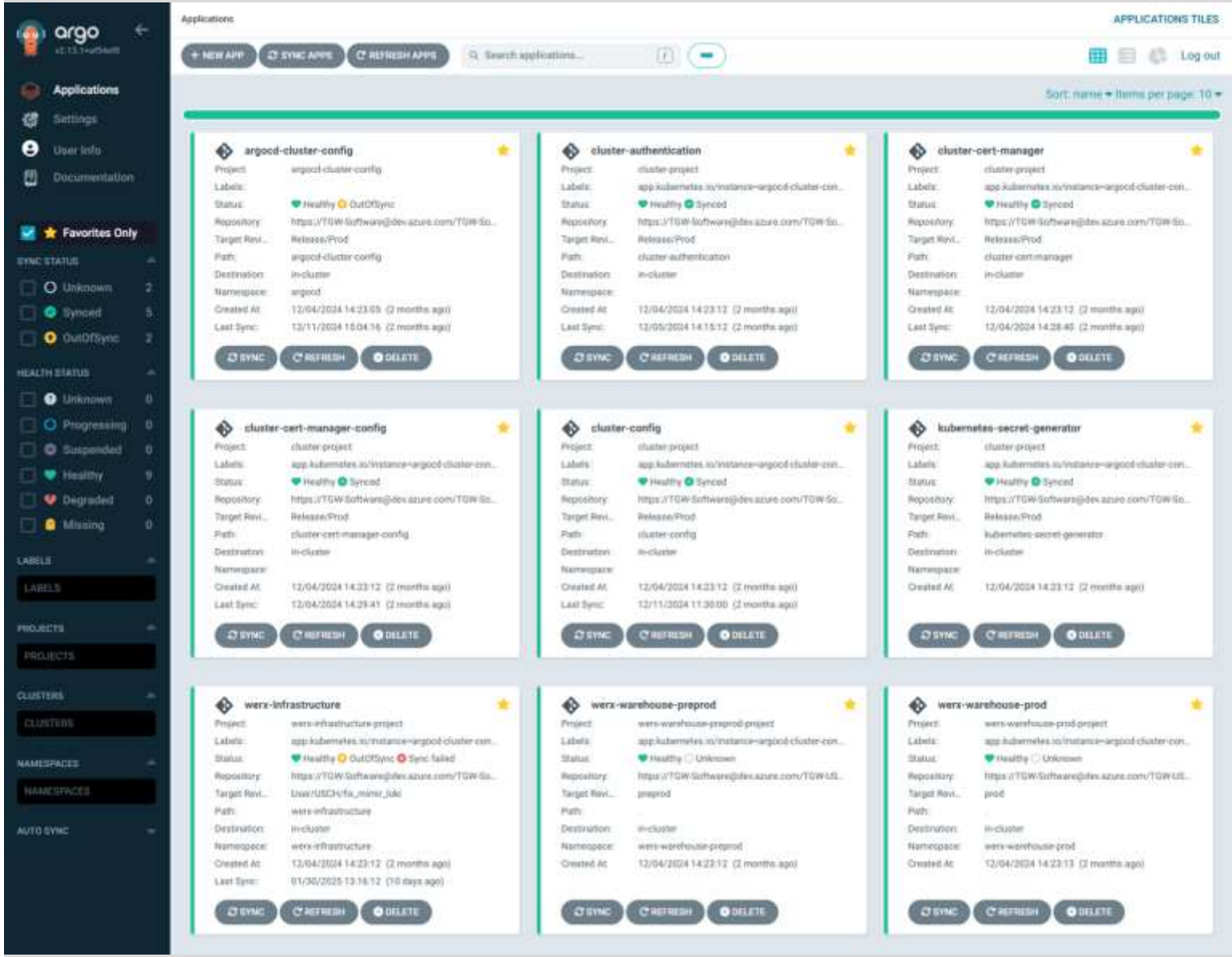
post OpenShift Installation there‘s one script to install argocd and setup / configure declarative gitops.



Installation

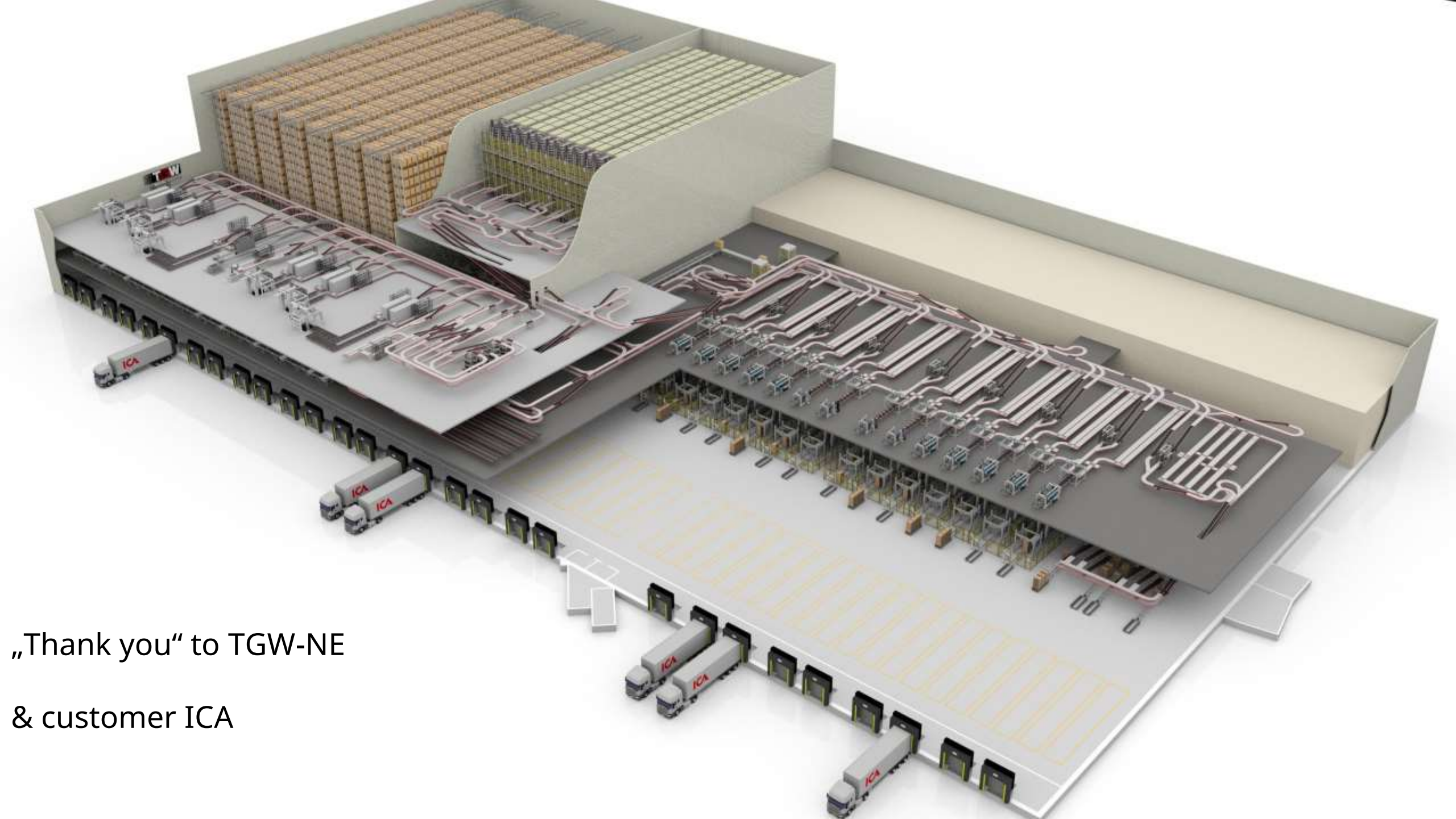
In argocd app-of-apps we have

- Cluster config
- Authentication
- Cert-manager installation / config
- Cluster wide daemonsets (logging)
- Prod / pre-prod applications



how we use openshift & argo

(deep dive into an actual customer project)



„Thank you“ to TGW-NE

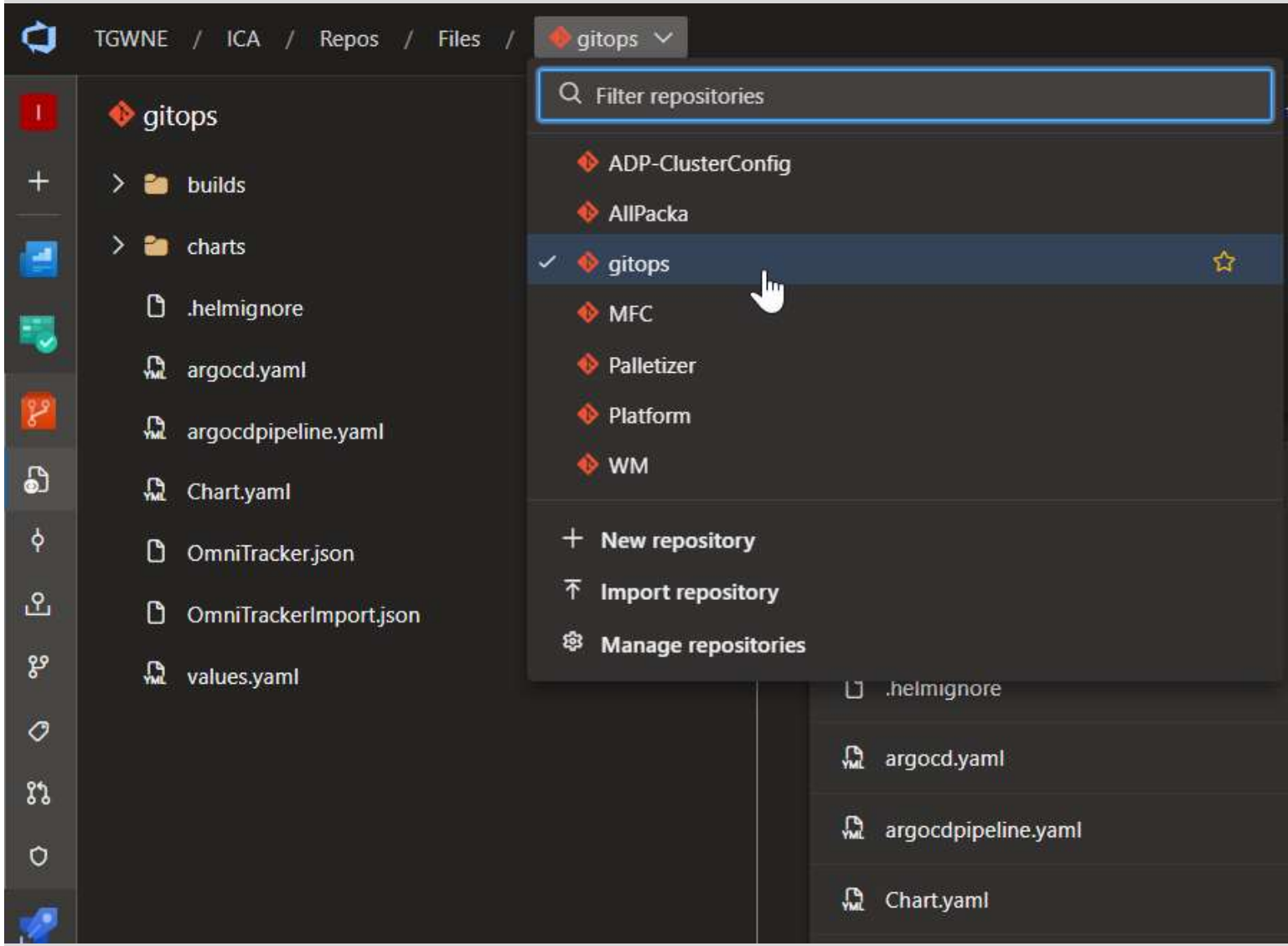
& customer ICA

How we use openshift

Each warehouse / each customer project has its own azure devops project.

In each project there are multiple repos.

- One for each component that is required,
- One „ADP-ClusterConfig“ repo containing,
- One „gitops“ repo containing kubernetes definitions for all applications



How we use openshift

Projects usually start in Azure DevOps:

We have s self service pipeline devs can go to.

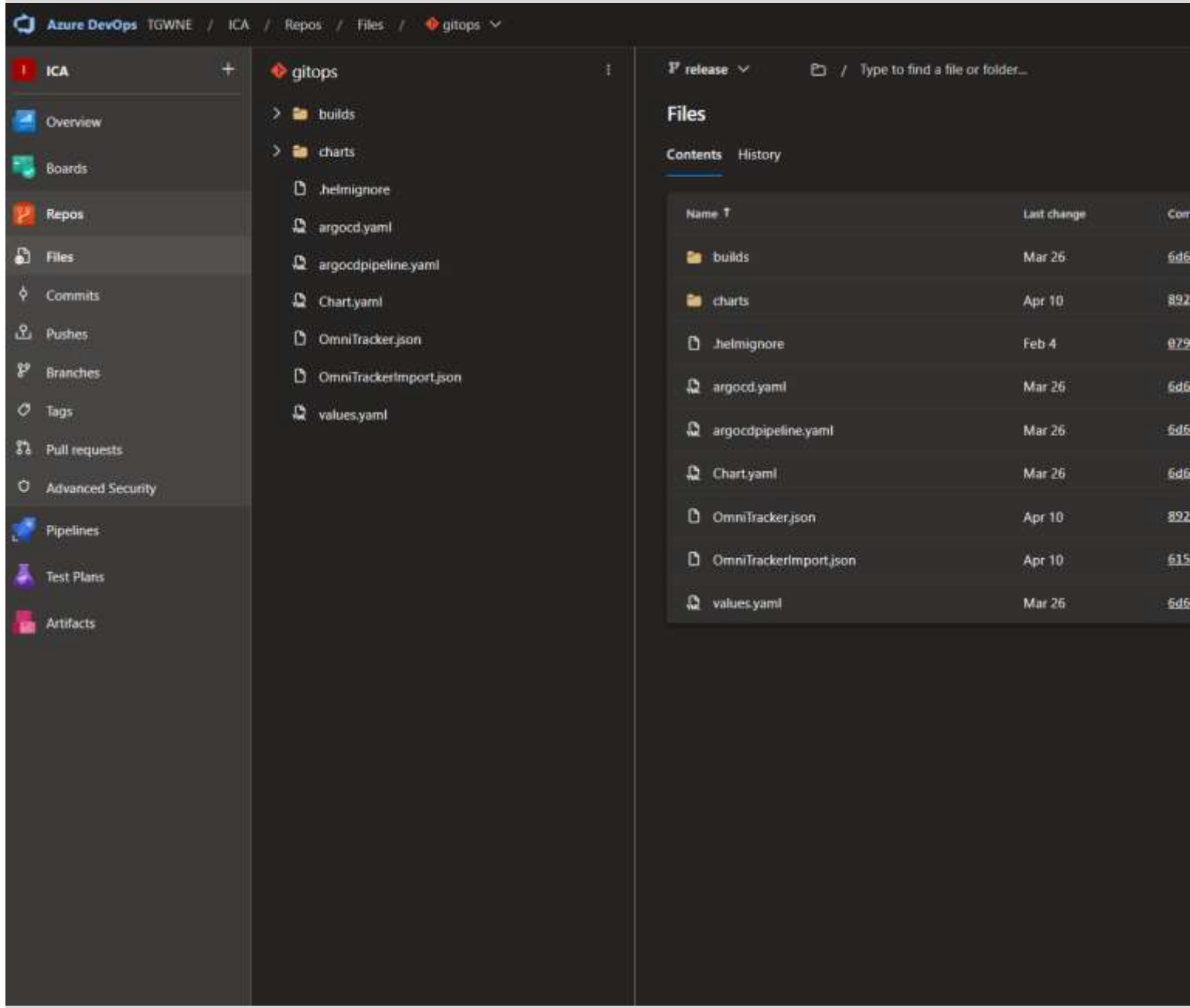
- It creates a new devops project…
- clones all desired component / product releases…
- adds azure devops pipelines…
- sets up links / permissions for container registries…
- sets up permissions for repos / build user…
- applies basic configuration (org specific feeds,…)

After ~10minutes project is set up in clean and standardized way.

How we use openshift

gitops repo essentially is one large HELM umbrella chart, containing (almost*) all information required to run a site.

*exceptions: production secrets that should not go into source control (oracle connection strings, cert-manager secrets)

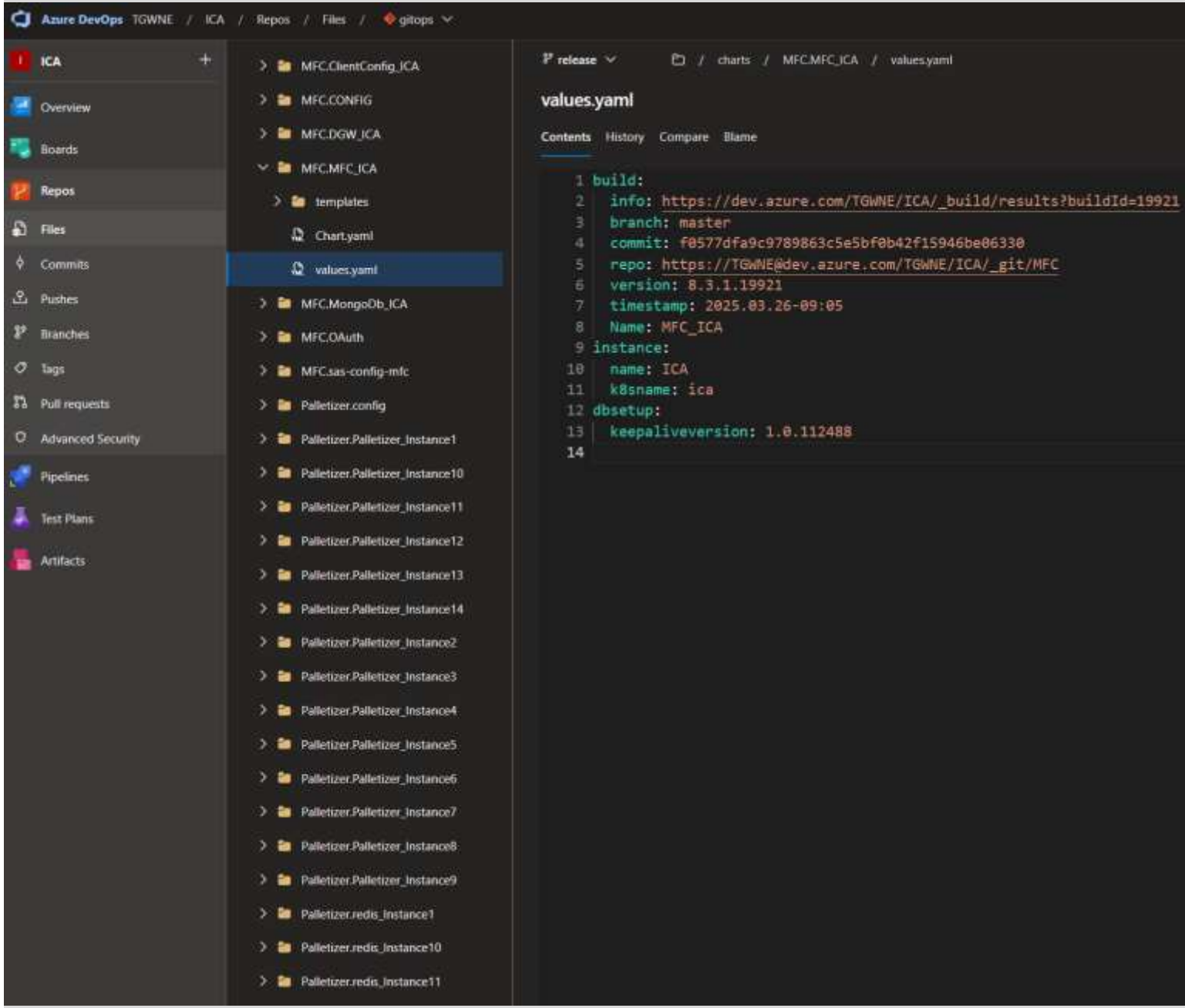


How we use openshift

Ground rules we have that show up in gitops repo

- New deliveries to customer environments can be staged.
- Publishing from stage to production must not rebuild
- None of our cluster is reachable from internet
- We don‘t „push“ into the customers environment.
  Customer is in control - environment is pulling.*

* this helps a lot in customer discussions

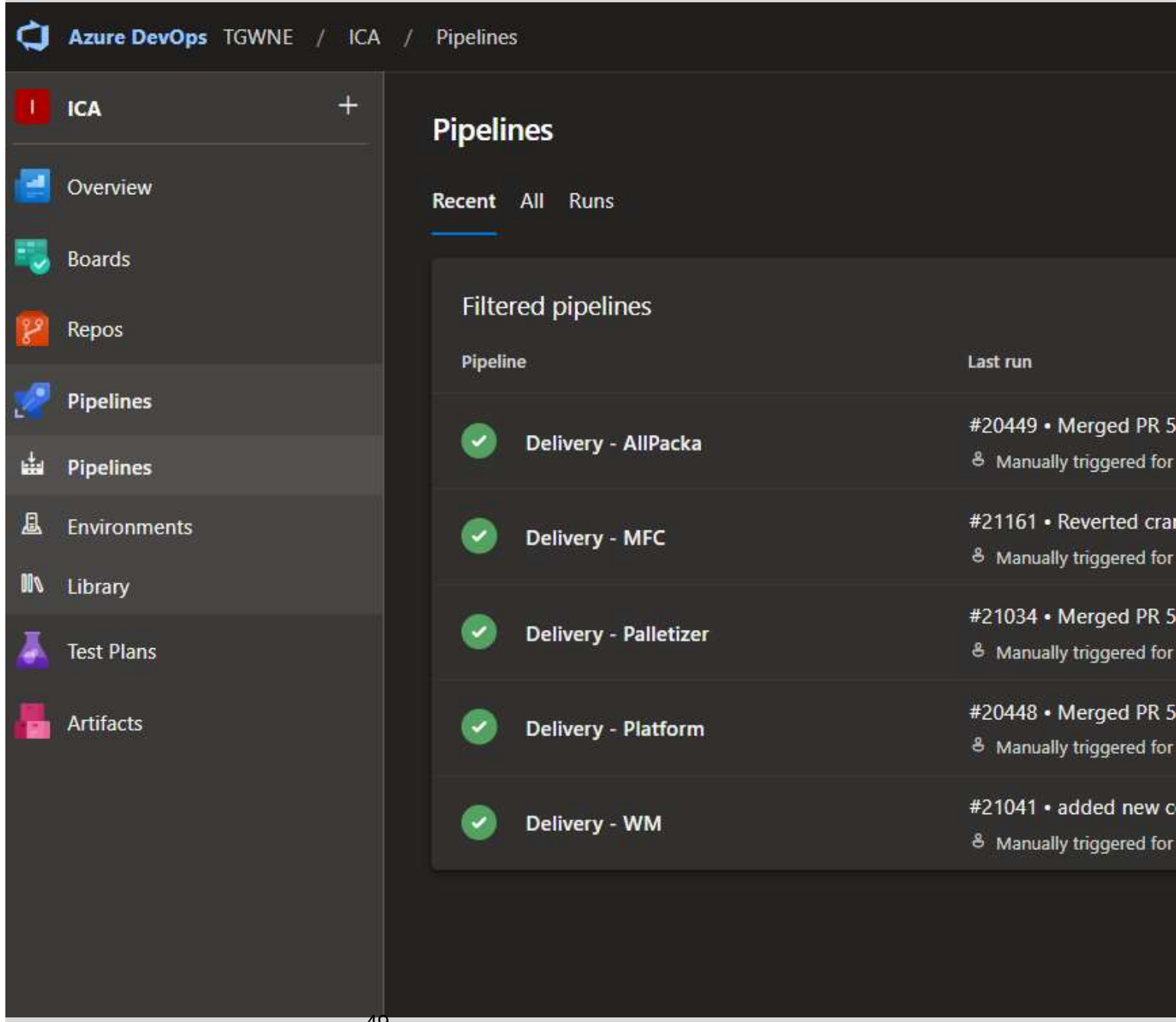


How we use openshift

Gitops repo is not manipulated directly / manually. This happens exlusively through pipelines*.

Each component has a „Delivery“ pipeline

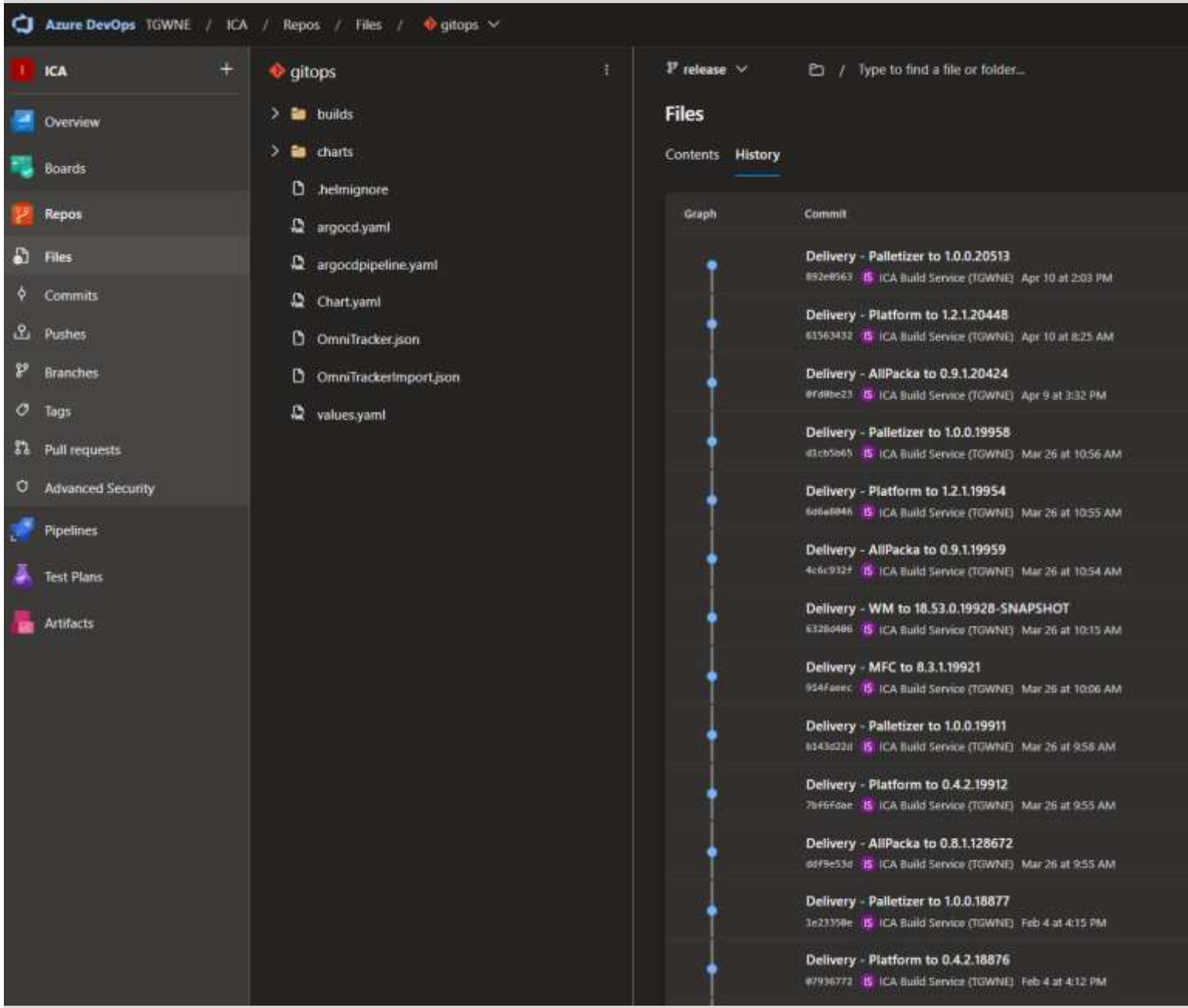
* At least for regular workflows and production use cases. Can be different during development of kubernetes resources



How we use openshift

Delivery Pipelines...

- Compile source code,
- Embedd static configuration
- Other policies (code analysis, package/license scans,...)
- create SBOM files,
- create container images,
- push their kubernetes definitions into the gitops repo

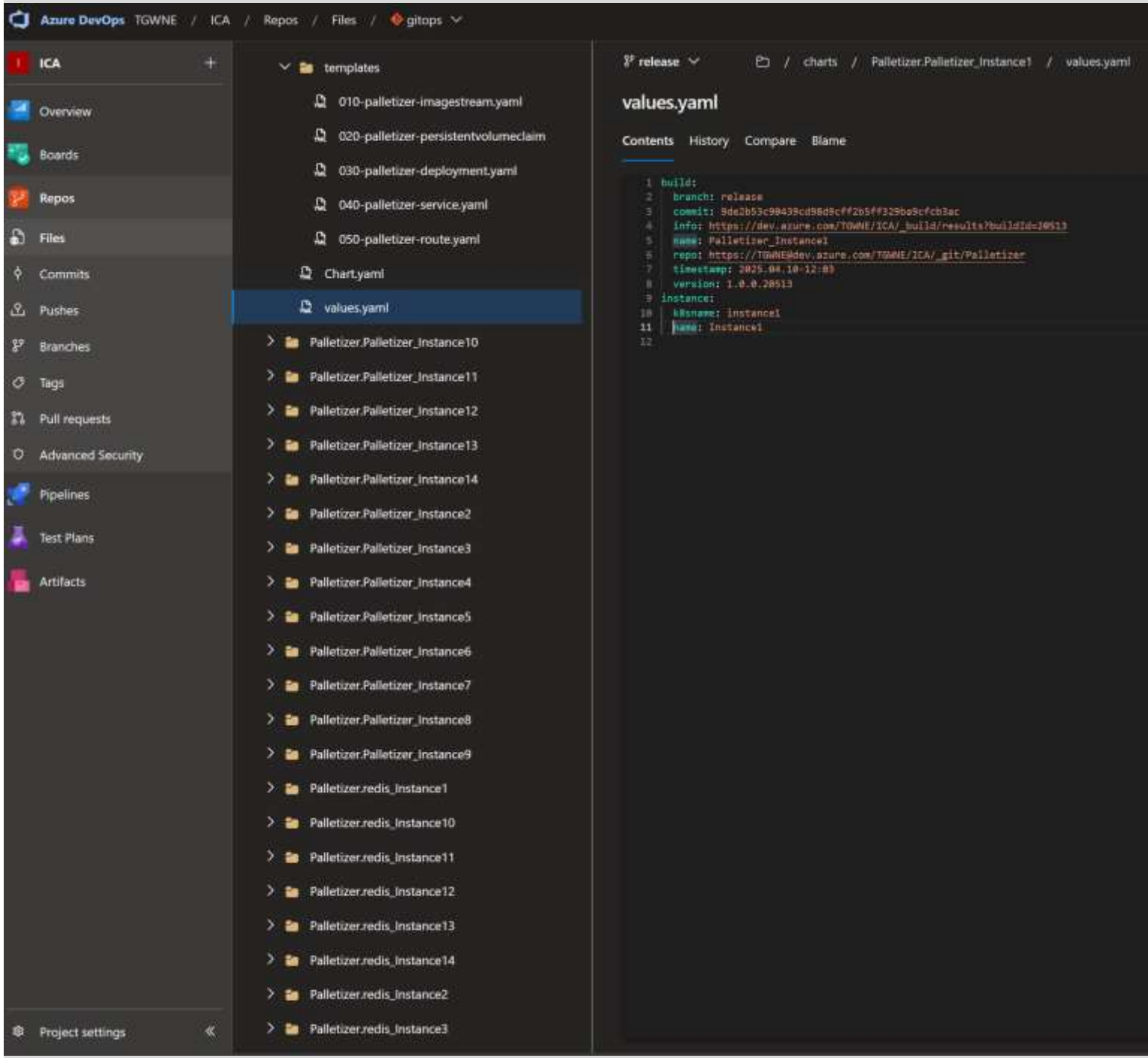


How we use openshift

We use kubernetes horizontal scaling – but not exclusively.

Most of our scaling requirements are bound to the underlying devices.

If we have f.e. 14 palletizer devices running in a customer environment, we have 14 instances running (each one connecting to a specific device and controlling it tightly)

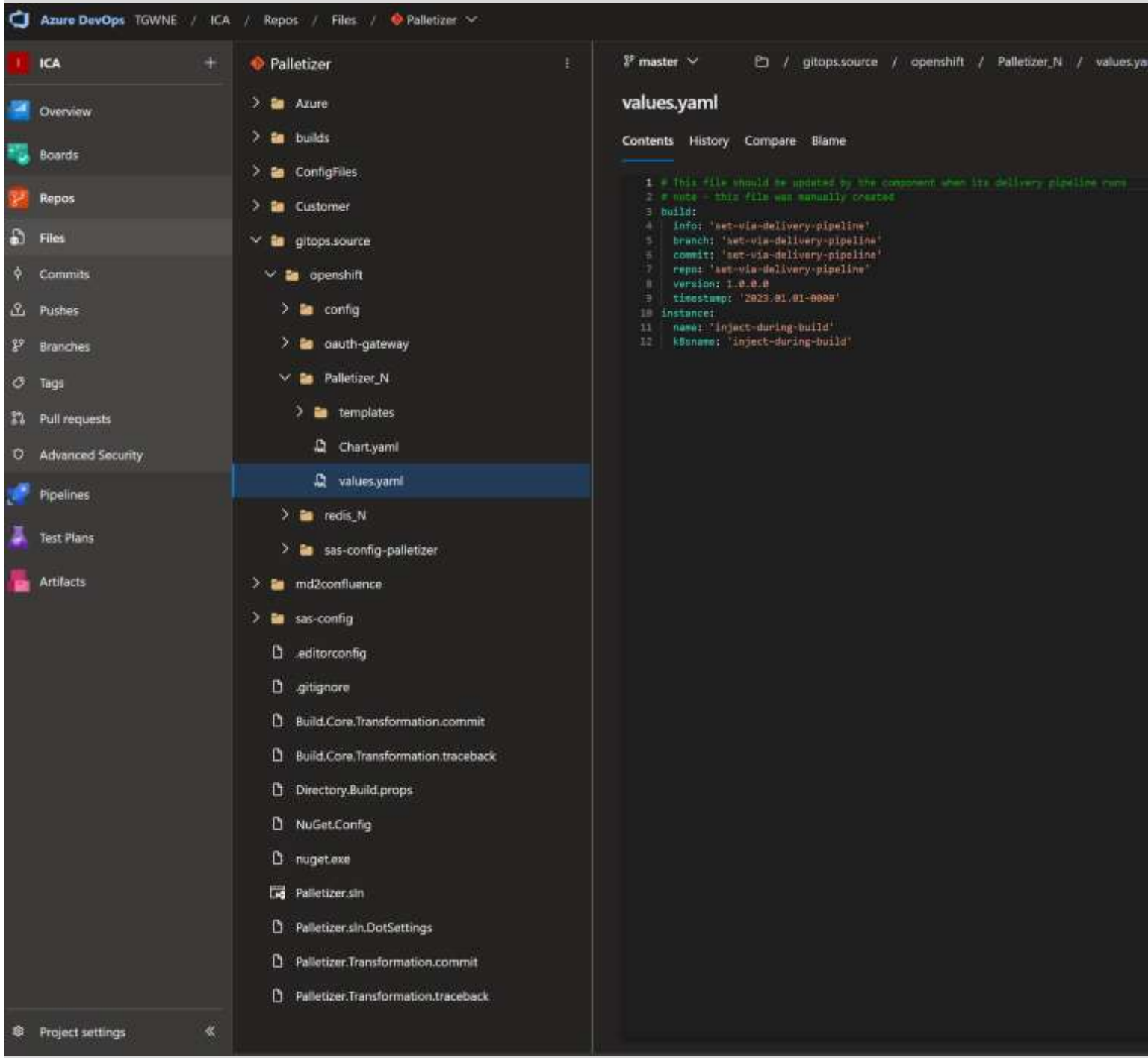


How we use openshift

Internally this is done by the Delivery Pipelines

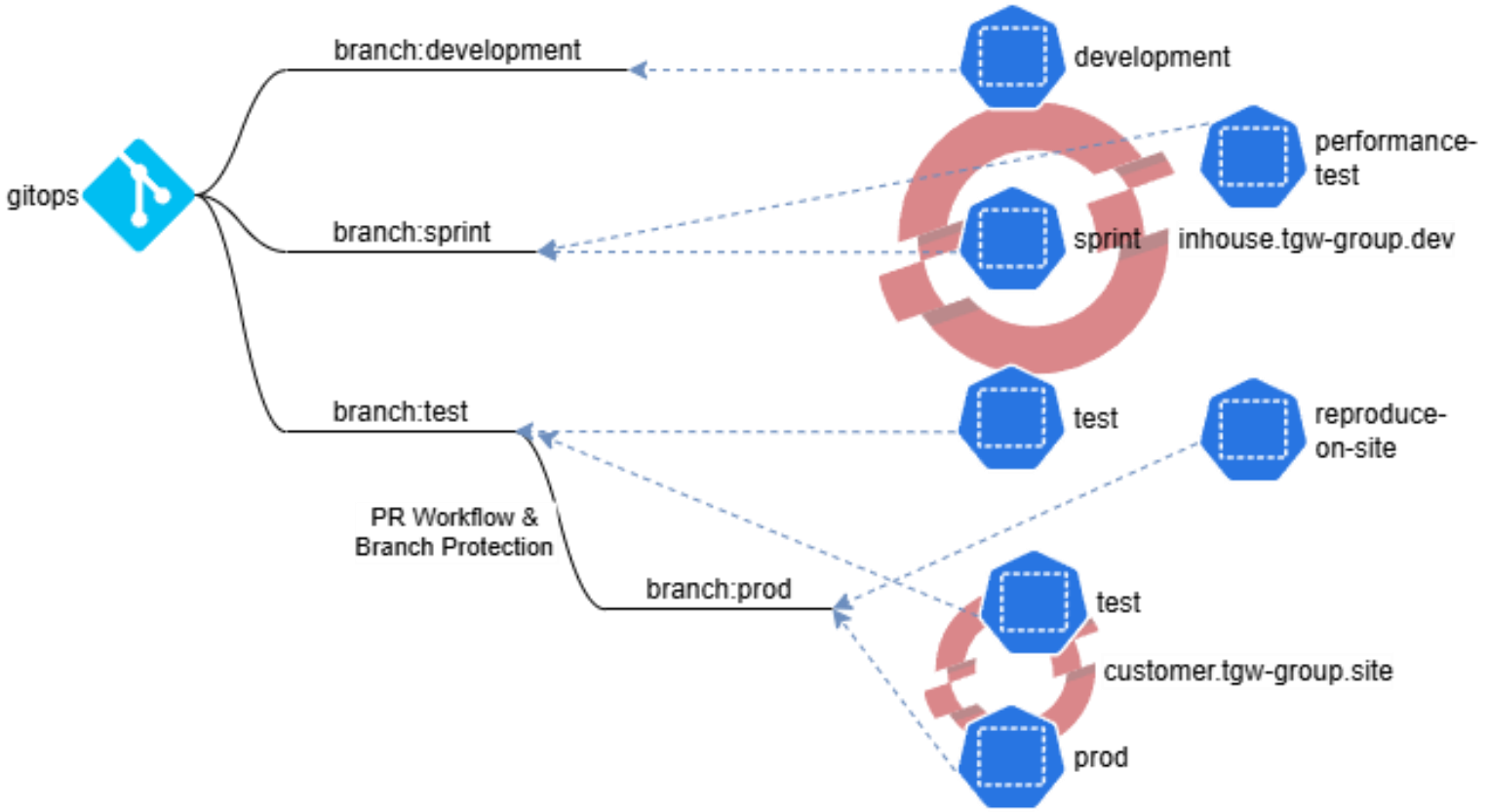
The source components repo has only one single palletizer kubernetes definition. The delivery pipeline checks the config how many instances are configured and expands accordingly.

This required additional effort on our side during pipeline creation, however we can control each device individually (and even update them individually)



How we use openshift

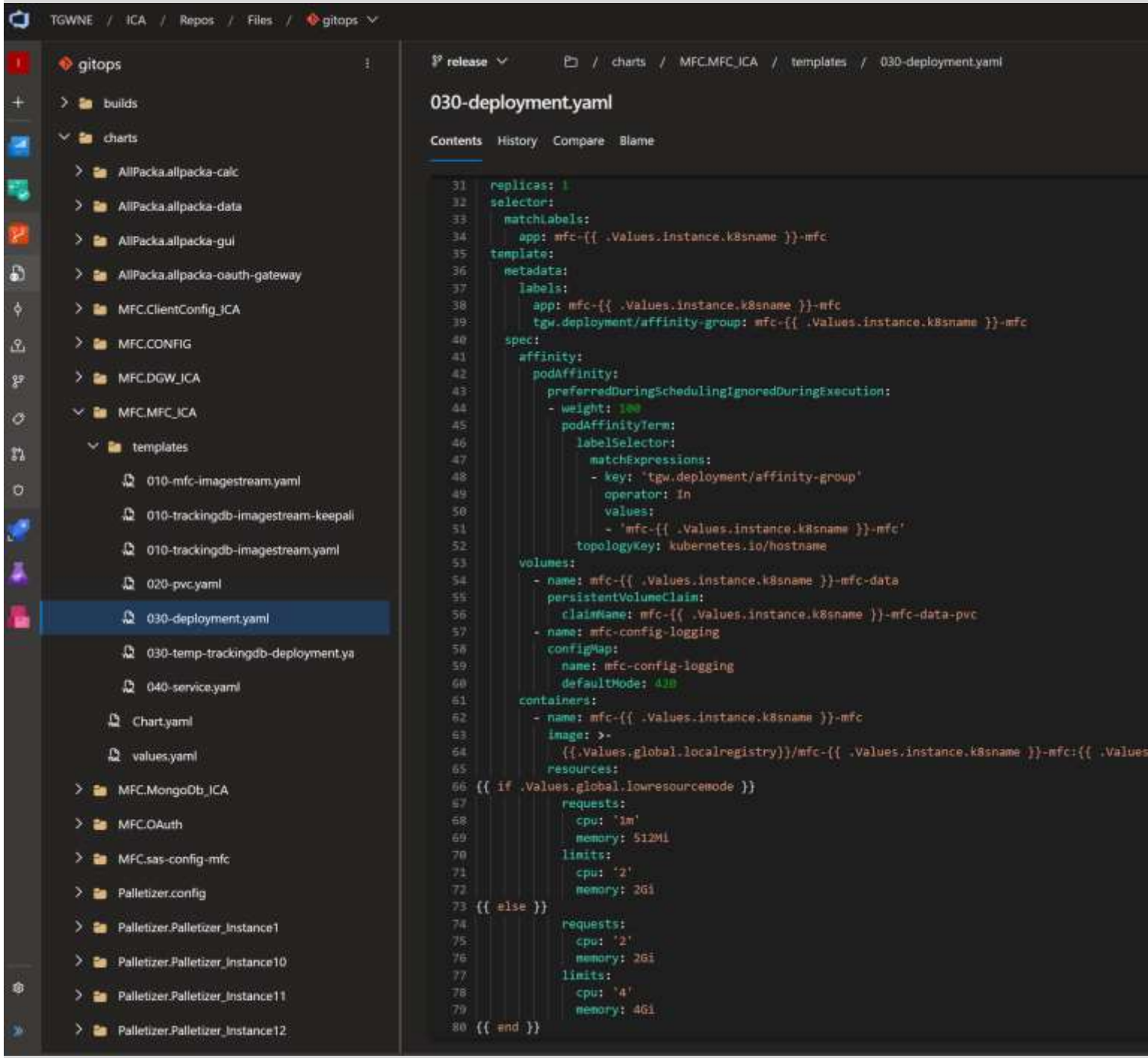
A single branch in the gitops repo can run in multiple namespaces in multiple clusters.



How we use openshift

A single branch in the gitops repo can run in multiple namespaces in multiple clusters.

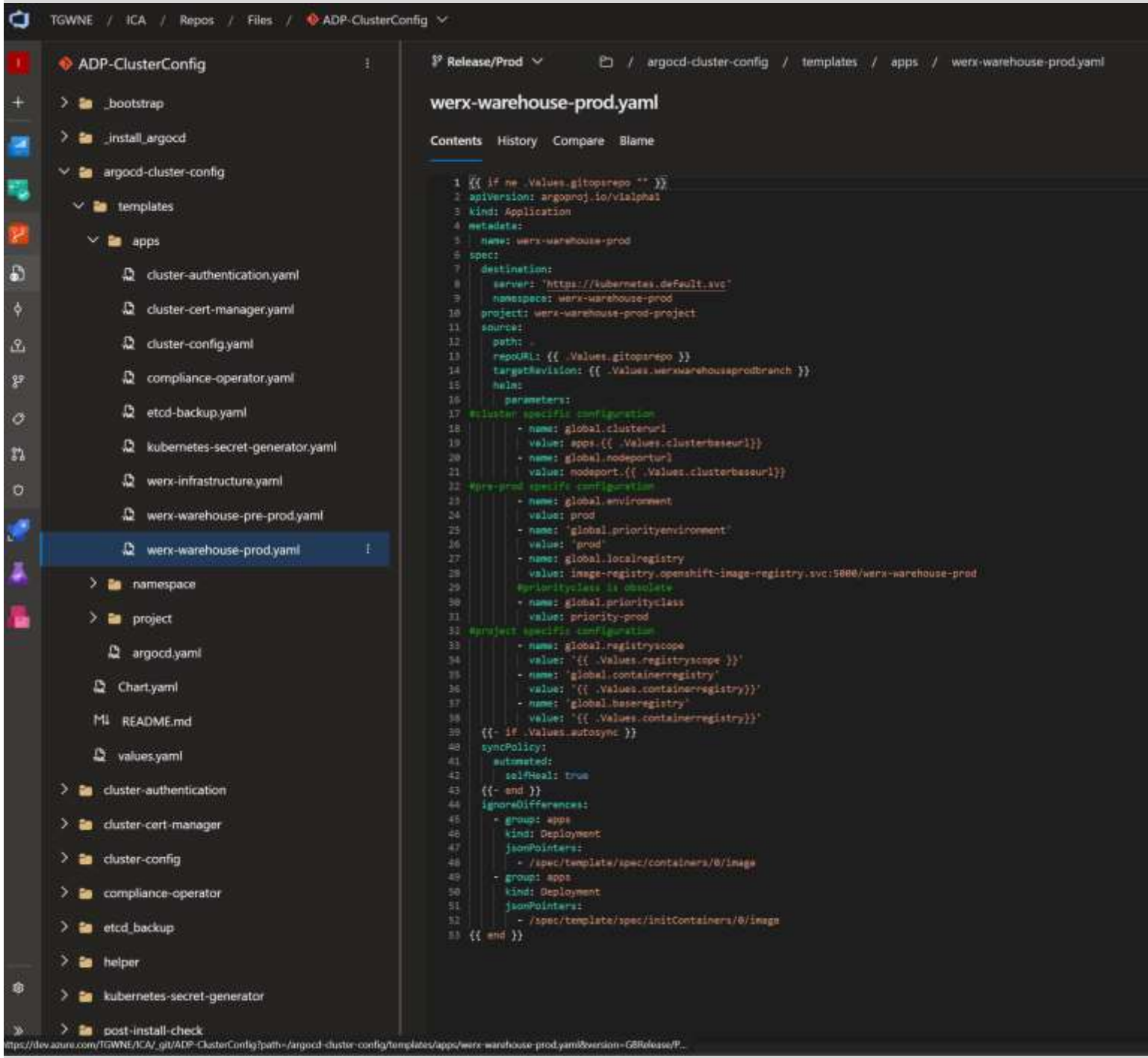
For this we‘re using lots of HELM variables in our kubernetes definitions that we‘re providing via argocd: cluster-url, prod/pre-prod/dev switch, …



How we use openshift

In production environments those variables are passed through the argocd app-of-apps coming from the ADP-ClusterConfig repo.

(And the argocd app-of-apps gets some of the values from the installation script)



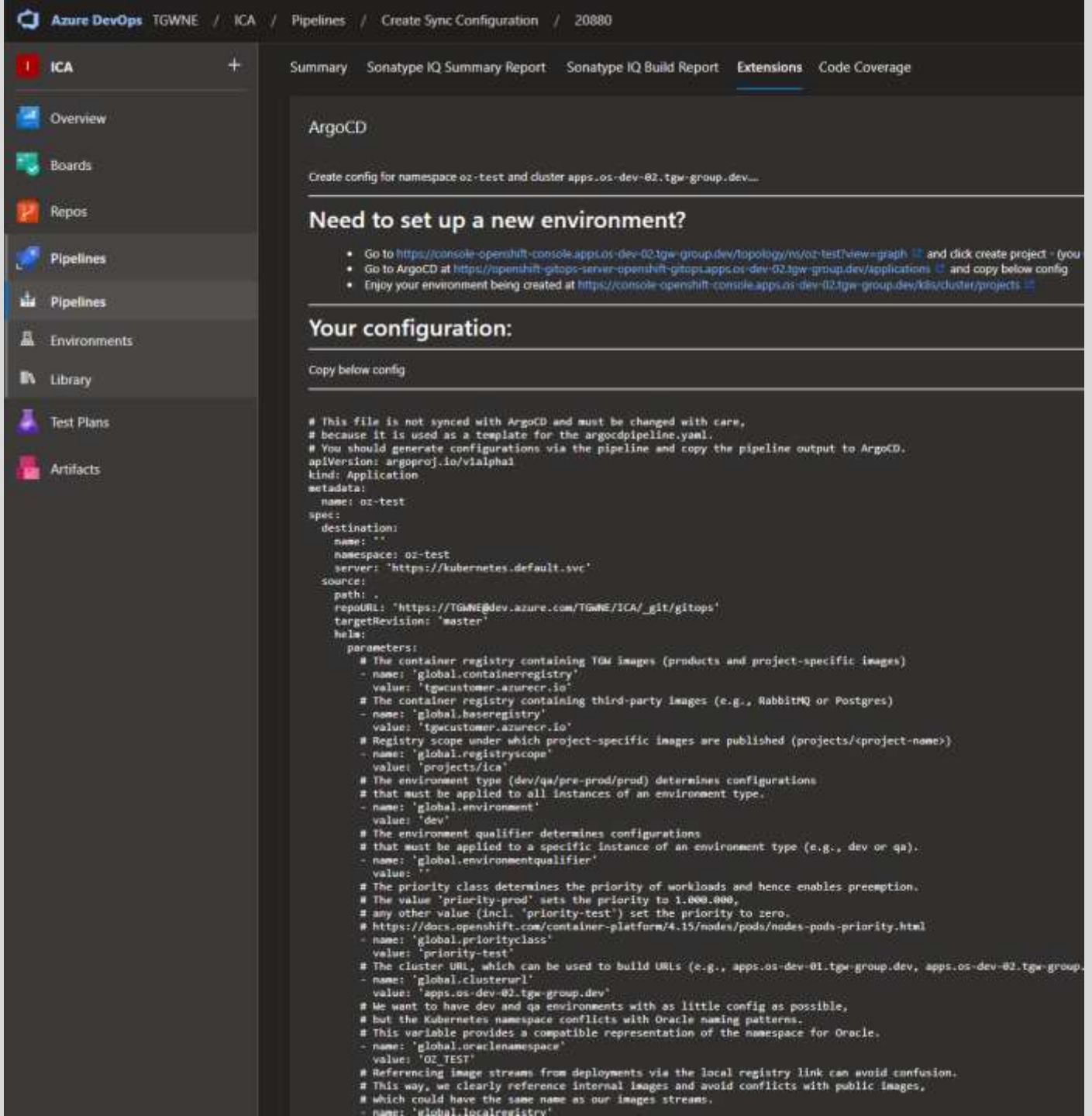
How we use openshift

Our internal test environments look very similar, but…

The app-of-apps and the cluster configs have HELM switches in it as well. If we have the inhouse flag, argocd allows self-service – developers can provision their environments themselfes*.

To set this up each project has a pipeline. Fill in the target namespace you want to set up and you‘ll get the argocd configuration.

*At the moment manually – as we‘re dealing with lots of different organizations.



How we use openshift

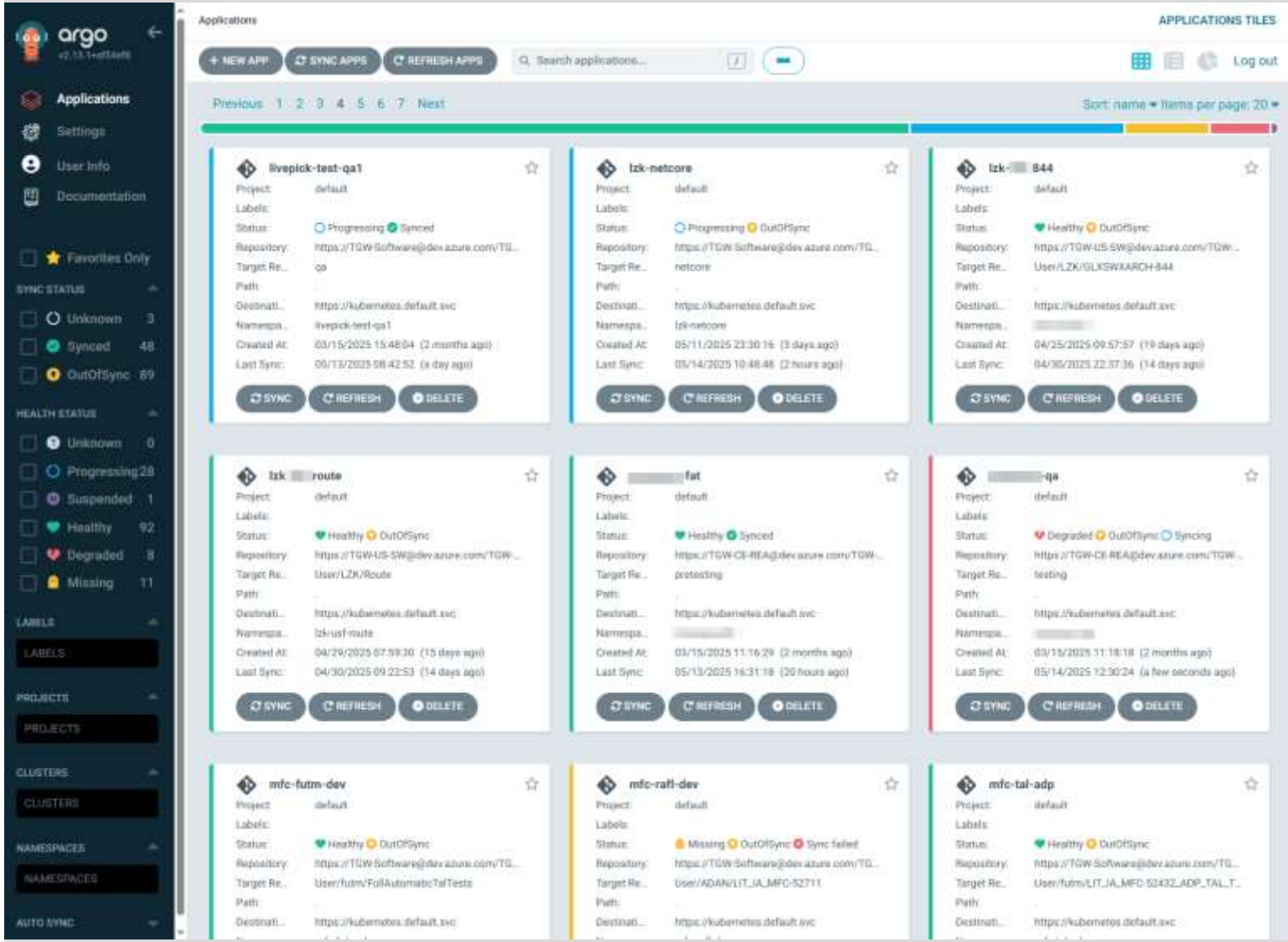
Internally we have 2 main clusters:

dev for regular day to day testing of new features. Around 120-140 full environments configured; spread out over around 25 nodes (usually 8 CPU, 32GB RAM) in multiple datacenter.

Only ~40 able to run concurrently.

On the one hand: CSI driver limits (60/node)

On the other hand: resource constraints *for development* are hard to estimate / set.
CPU is never a problem.
RAM is hard. Basic functional tests are OK.
But as soon as some load goes into it, eviction errors cascade.

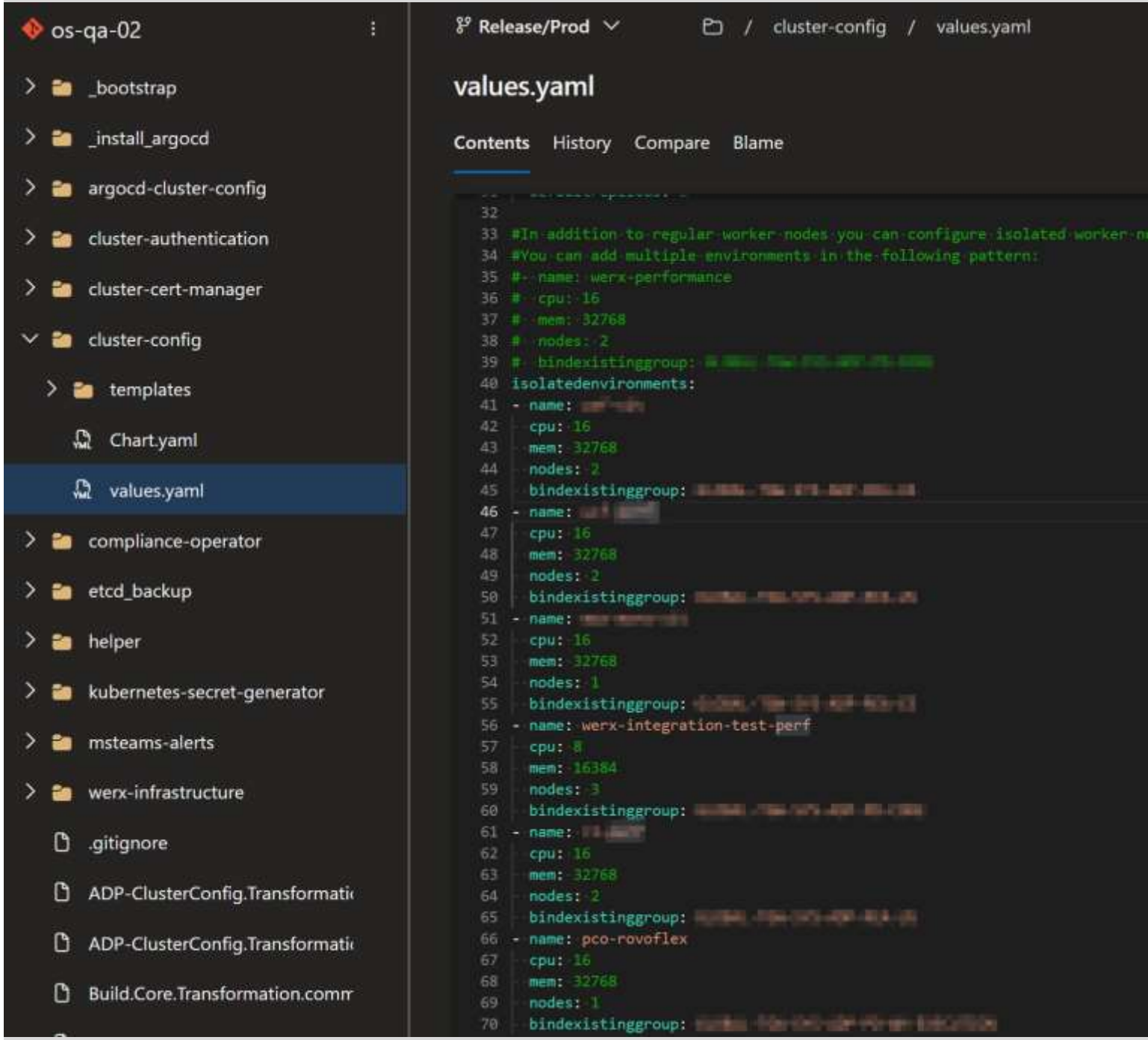


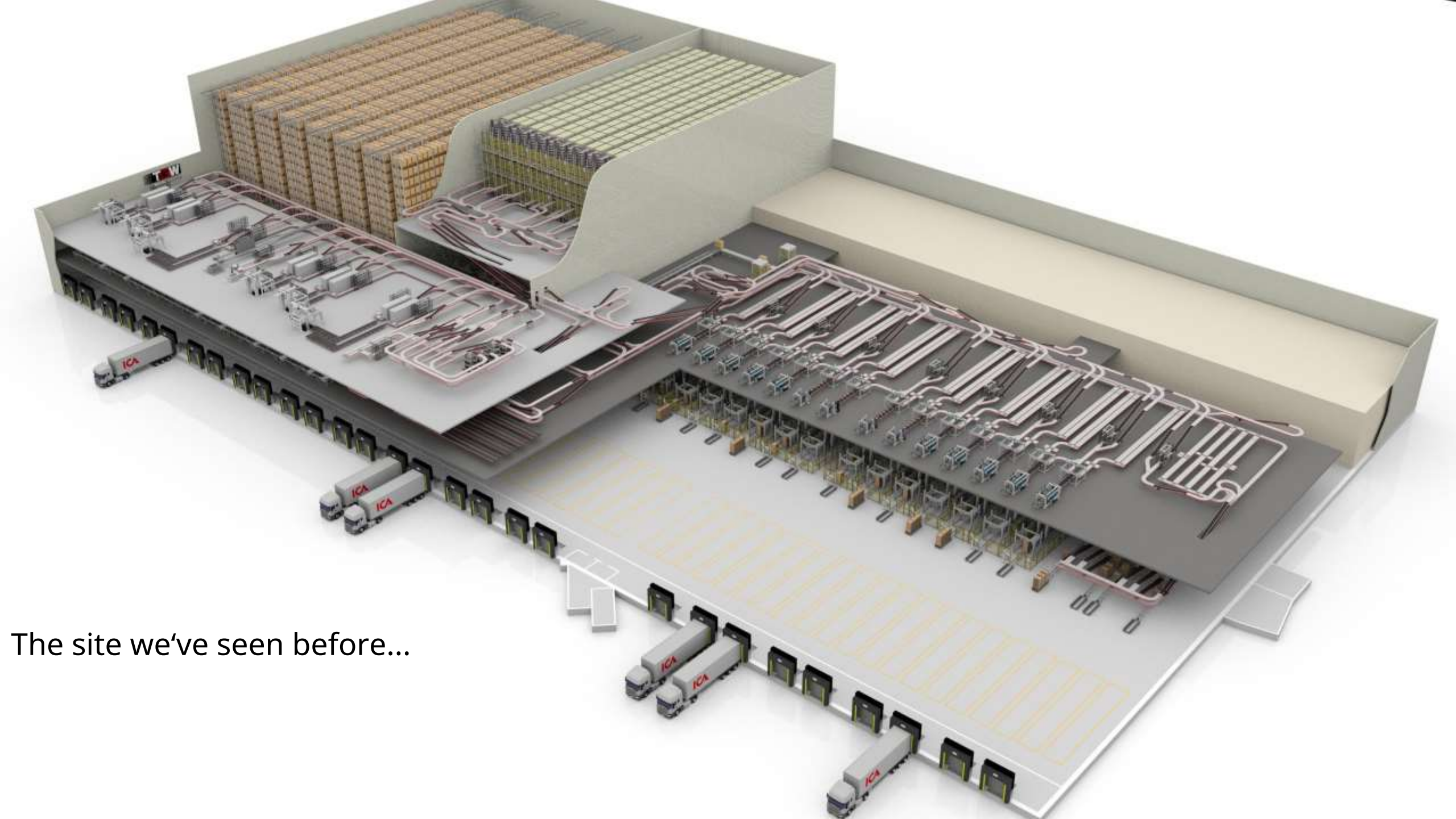
How we use openshift

Internally we have 2 main clusters:

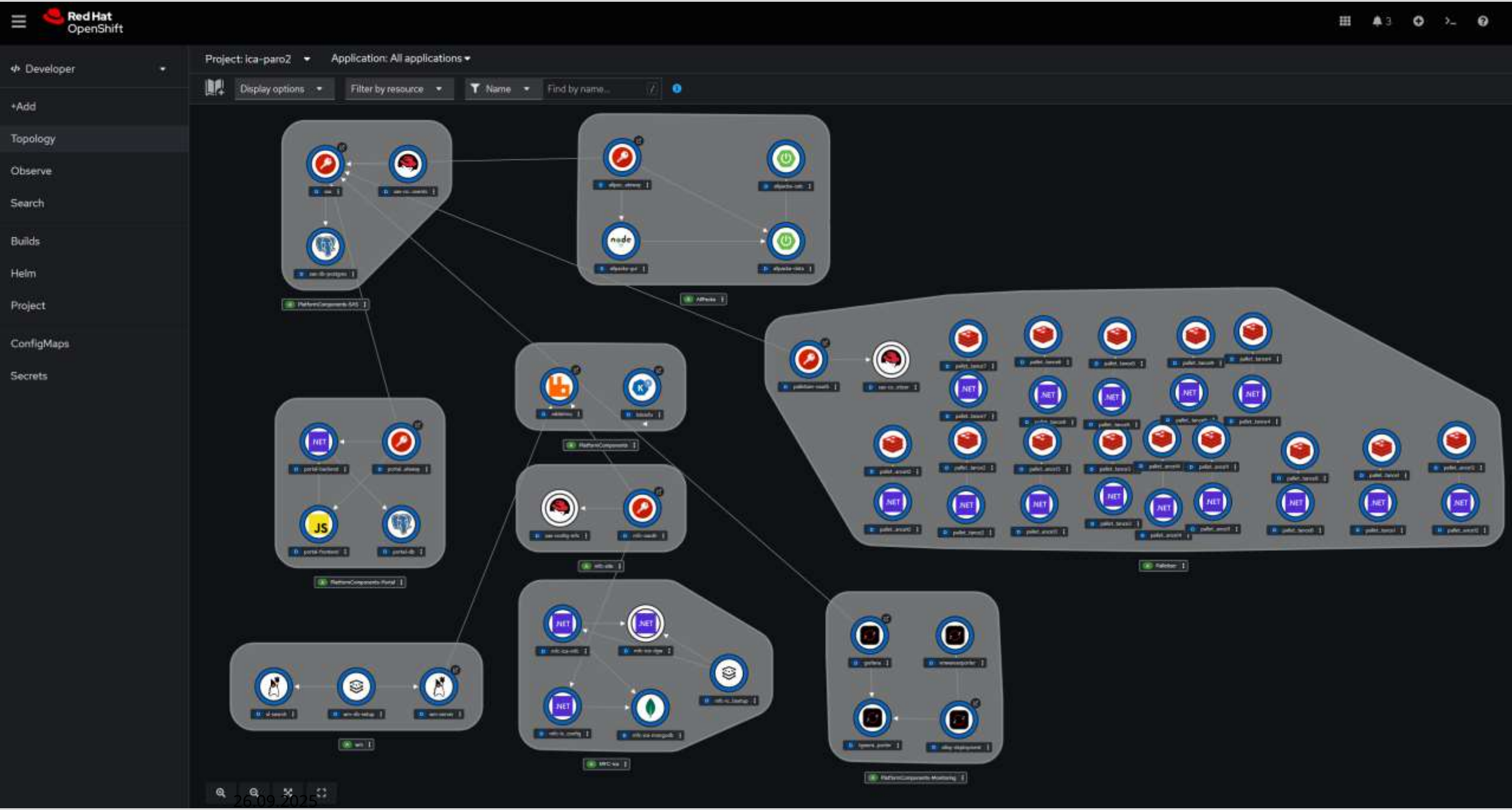
qa for special load tests where you need to have a „controlled“ environment that is compareable
(is my application now slower because of a bug or because there‘s additional load on the cluster).

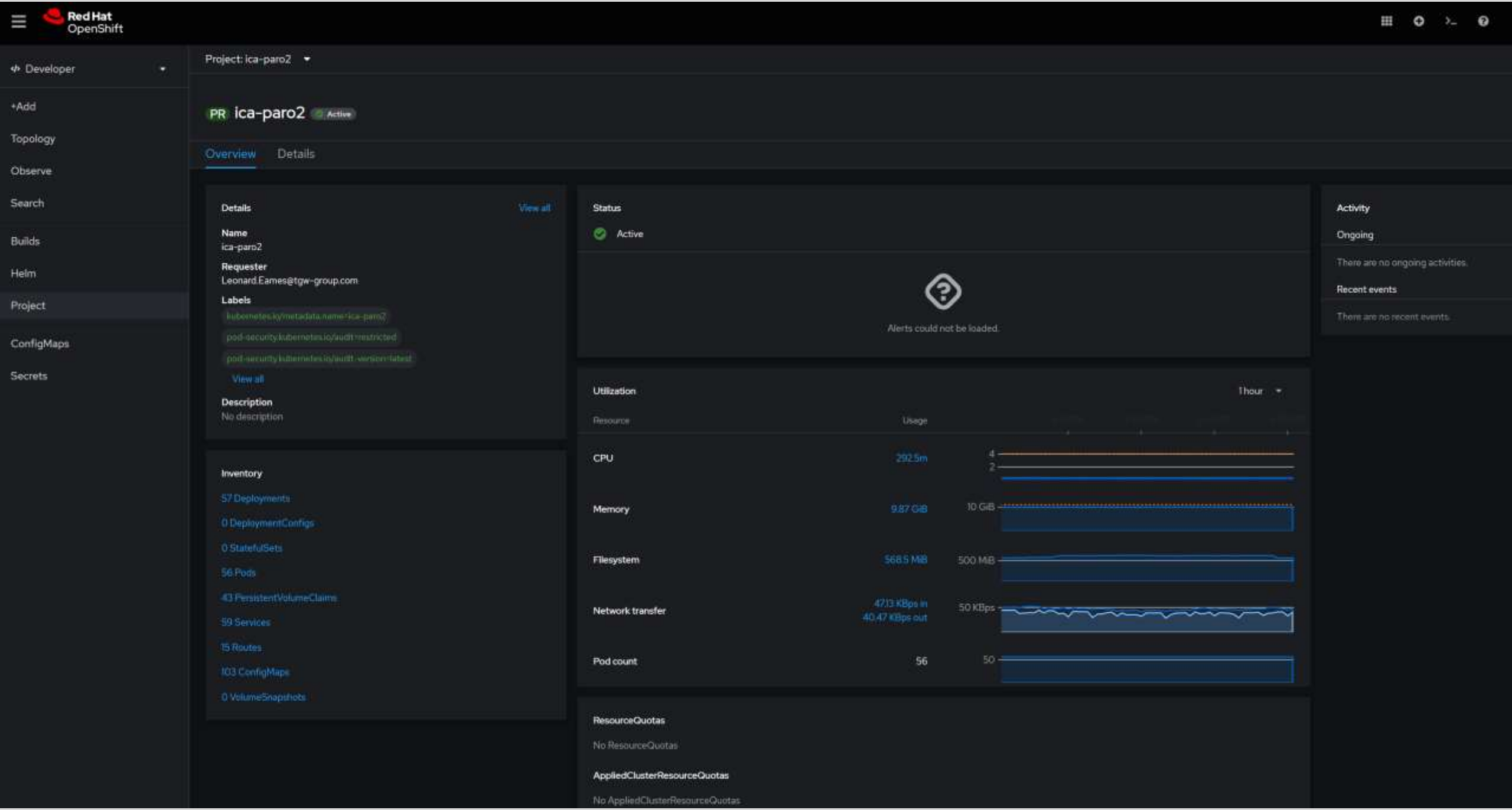
Each namespace goes to dedicated nodes.





The site we‘ve seen before…


Red Hat OpenShift
Developer
+Add
Topology
Observe
Search
Builds
Helm
Project
ConfigMaps
Secrets
Project: ica-paroZ
Application: All applications
Display options
Filter by resource
Name
Find by name...
node
JS
.NET


Red Hat OpenShift
Developer
Project: ica-paro2
+Add
Topology
Observe
Search
Builds
Helm
Project
ConfigMaps
Secrets
PR ica-paro2
Active
Overview
Details
Details
View all
Name
ica-paro2
Requester
Leonard.Eames@tgw-group.com
Labels
kubernetes.io/metadata.name=ica-paro2
pod-security.kubernetes.io/audit=restricted
pod-security.kubernetes.io/audit-version=latest
View all
Description
No description
Inventory
57 Deployments
0 DeploymentConfigs
0 StatefulSets
56 Pods
43 PersistentVolumeClaims
59 Services
15 Routes
103 ConfigMaps
0 VolumeSnapshots
Status
Active
Alerts could not be loaded.
Utilization
1 hour
Resource
Usage
CPU
292.5m
4
2
Memory
9.87 GiB
10 GiB
Filesystem
568.5 MiB
500 MiB
Network transfer
47.13 KBps in
40.47 KBps out
50 KBps
Pod count
56
50
ResourceQuotas
No ResourceQuotas
AppliedClusterResourceQuotas
No AppliedClusterResourceQuotas
Activity
Ongoing
There are no ongoing activities.
Recent events
There are no recent events.

So – how does all of this improve the initial setup?

Change

Before we started, setting up a new inhouse environment took a week of skilled effort.

Now it takes 10minutes:
1 minute of work, 9 minutes of waiting

(which is awesome the first few times you see deployment after deployment popping up in the Topology view)

Environments can be started/stopped on demand – as it can be automated entirely.

challenges

after 3 years inhouse

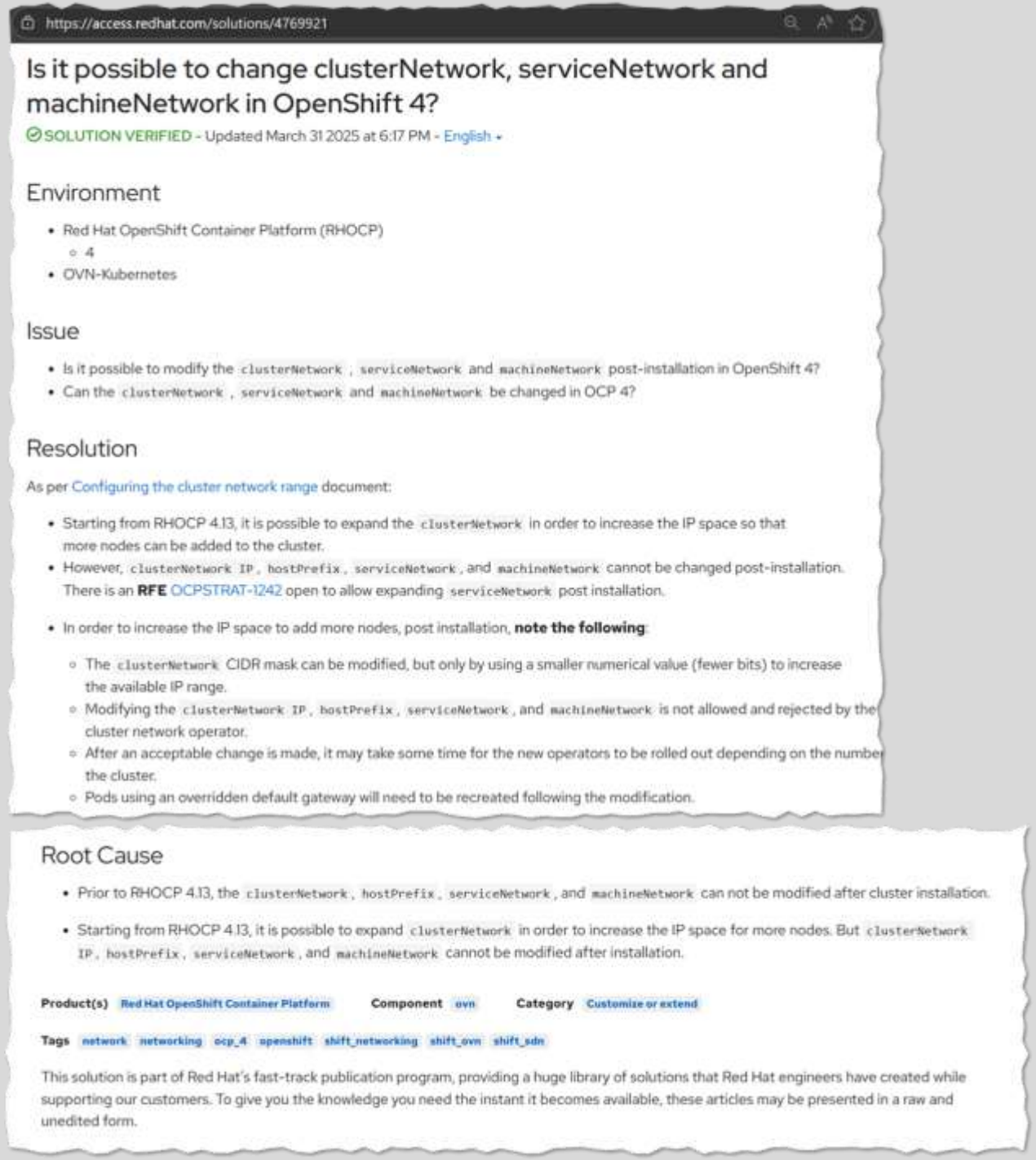
after 1 year at customer

Challenges Network

When we install in a new customer warehouse, it is common that network is set up just prior.

Customer regularly wants to update network configuration (new IP addresses for master/ingress/api, change from DHCP to static or vice versa).

This in itself is no unreasonable request. However it's one that we usually push off as risks are too high.



https://access.redhat.com/solutions/4769921

Is it possible to change clusterNetwork, serviceNetwork and machineNetwork in OpenShift 4?

SOLUTION VERIFIED - Updated March 31 2025 at 6:17 PM - English

Environment

- Red Hat OpenShift Container Platform (RHOCP)
  - 4
- OVN-Kubernetes

Issue

- Is it possible to modify the `clusterNetwork` , `serviceNetwork` and `machineNetwork` post-installation in OpenShift 4?
- Can the `clusterNetwork` , `serviceNetwork` and `machineNetwork` be changed in OCP 4?

Resolution

As per Configuring the cluster network range document:

- Starting from RHOCP 4.13, it is possible to expand the `clusterNetwork` in order to increase the IP space so that more nodes can be added to the cluster.
- However, `clusterNetwork IP` , `hostPrefix` , `serviceNetwork` , and `machineNetwork` cannot be changed post-installation. There is an **RFE** OCPSTRAT-1242 open to allow expanding `serviceNetwork` post installation.
- In order to increase the IP space to add more nodes, post installation, **note the following**:
  - The `clusterNetwork` CIDR mask can be modified, but only by using a smaller numerical value (fewer bits) to increase the available IP range.
  - Modifying the `clusterNetwork IP` , `hostPrefix` , `serviceNetwork` , and `machineNetwork` is not allowed and rejected by the cluster network operator.
  - After an acceptable change is made, it may take some time for the new operators to be rolled out depending on the number the cluster.
  - Pods using an overridden default gateway will need to be recreated following the modification.

Root Cause

- Prior to RHOCP 4.13, the `clusterNetwork` , `hostPrefix` , `serviceNetwork` , and `machineNetwork` can not be modified after cluster installation.
- Starting from RHOCP 4.13, it is possible to expand `clusterNetwork` in order to increase the IP space for more nodes. But `clusterNetwork IP` , `hostPrefix` , `serviceNetwork` , and `machineNetwork` cannot be modified after installation.

Product(s) Red Hat OpenShift Container Platform **Component** ovn **Category** Customize or extend

Tags network networking ocp_4 openshift shift_networking shift_ovn shift_sdn

This solution is part of Red Hat's fast-track publication program, providing a huge library of solutions that Red Hat engineers have created while supporting our customers. To give you the knowledge you need the instant it becomes available, these articles may be presented in a raw and unedited form.

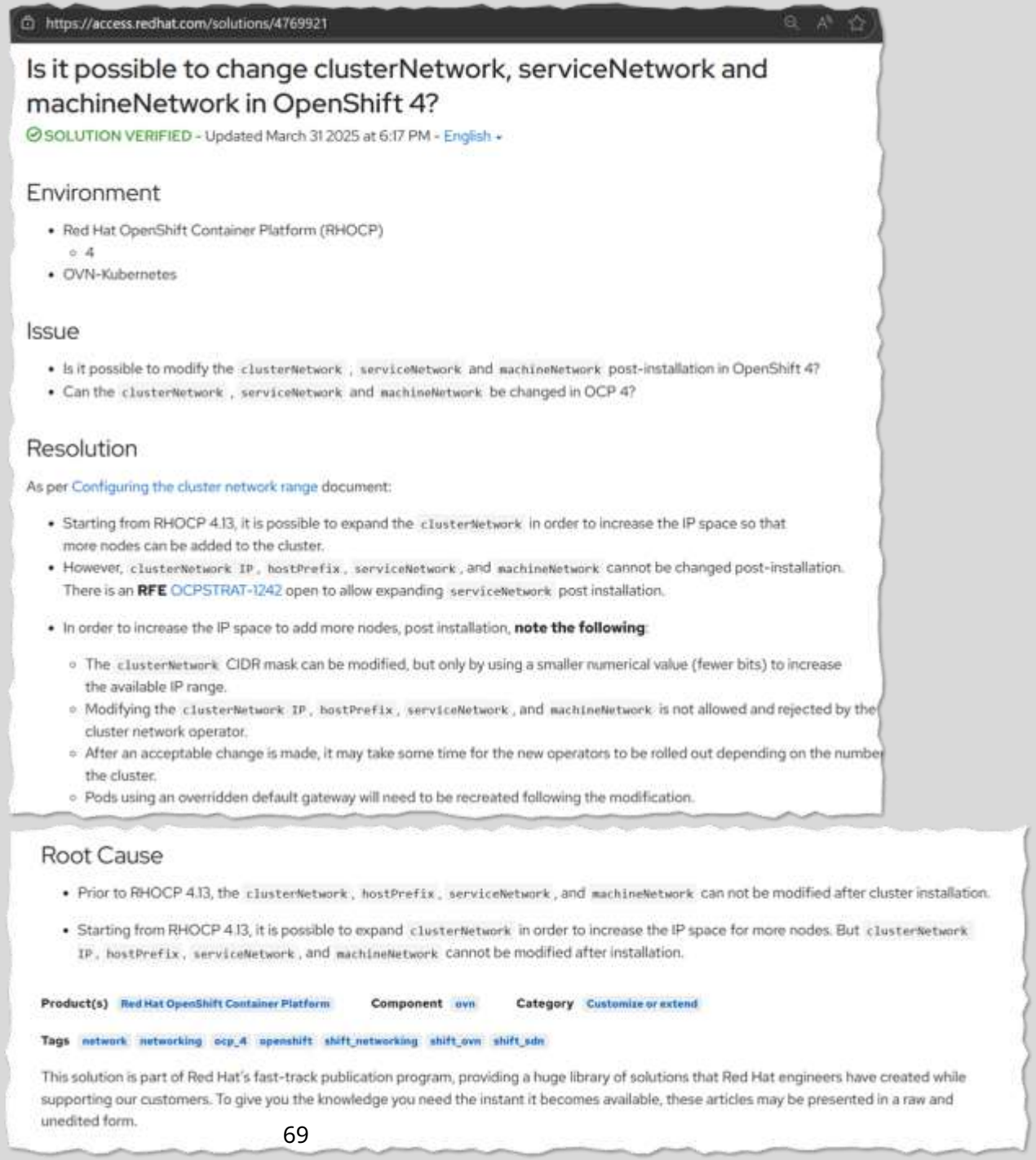
Challenges Network

Bites us internally as well…

The current dev cluster is a year old and should move to different servers.
Actual migration to servers is no problem (happens on VM Layer) – but this server has policy to use different VLANs.

No one dares to migrate there, so we need to install a new dev system, migrate applications, kill the old one.

Not rocket science, no live migration needed. But – effort for 100 people.



https://access.redhat.com/solutions/4769921

Is it possible to change clusterNetwork, serviceNetwork and machineNetwork in OpenShift 4?

SOLUTION VERIFIED - Updated March 31 2025 at 6:17 PM - English

Environment

- Red Hat OpenShift Container Platform (RHOCP)
  - 4
- OVN-Kubernetes

Issue

- Is it possible to modify the `clusterNetwork` , `serviceNetwork` and `machineNetwork` post-installation in OpenShift 4?
- Can the `clusterNetwork` , `serviceNetwork` and `machineNetwork` be changed in OCP 4?

Resolution

As per Configuring the cluster network range document:

- Starting from RHOCP 4.13, it is possible to expand the `clusterNetwork` in order to increase the IP space so that more nodes can be added to the cluster.
- However, `clusterNetwork IP` , `hostPrefix` , `serviceNetwork` , and `machineNetwork` cannot be changed post-installation. There is an **RFE** OCPSTRAT-1242 open to allow expanding `serviceNetwork` post installation.
- In order to increase the IP space to add more nodes, post installation, **note the following**:
  - The `clusterNetwork` CIDR mask can be modified, but only by using a smaller numerical value (fewer bits) to increase the available IP range.
  - Modifying the `clusterNetwork IP` , `hostPrefix` , `serviceNetwork` , and `machineNetwork` is not allowed and rejected by the cluster network operator.
  - After an acceptable change is made, it may take some time for the new operators to be rolled out depending on the number the cluster.
  - Pods using an overridden default gateway will need to be recreated following the modification.

Root Cause

- Prior to RHOCP 4.13, the `clusterNetwork` , `hostPrefix` , `serviceNetwork` , and `machineNetwork` can not be modified after cluster installation.
- Starting from RHOCP 4.13, it is possible to expand `clusterNetwork` in order to increase the IP space for more nodes. But `clusterNetwork IP` , `hostPrefix` , `serviceNetwork` , and `machineNetwork` cannot be modified after installation.

Product(s) Red Hat OpenShift Container Platform **Component** ovn **Category** Customize or extend

Tags network networking ocp_4 openshift shift_networking shift_ovn shift_sdn

This solution is part of Red Hat's fast-track publication program, providing a huge library of solutions that Red Hat engineers have created while supporting our customers. To give you the knowledge you need the instant it becomes available, these articles may be presented in a raw and unedited form.

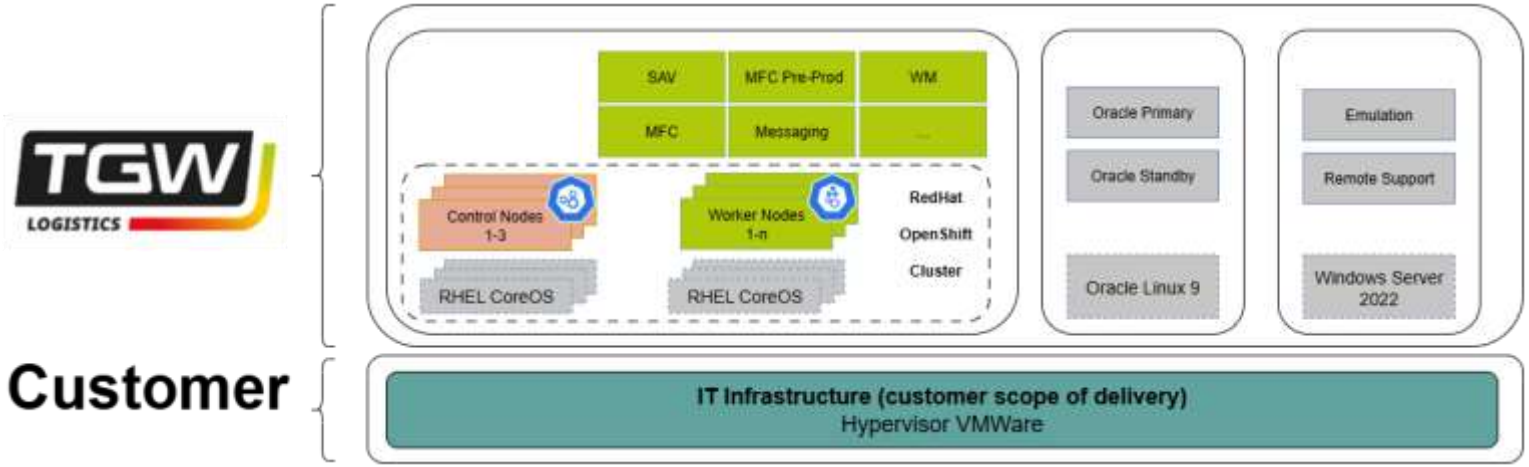
Challenges Storage

This is the biggest pain point for us at the moment – as it‘s the one we have least experience with:

Historically, we either ran black box were we had control of everything.

Or we ran grey box were customer was in control of everything – and we got storage through the hypervisor layer.

Customer could provision disks for us in the hypervisor, but we were not required to know/deal/manage storage details.

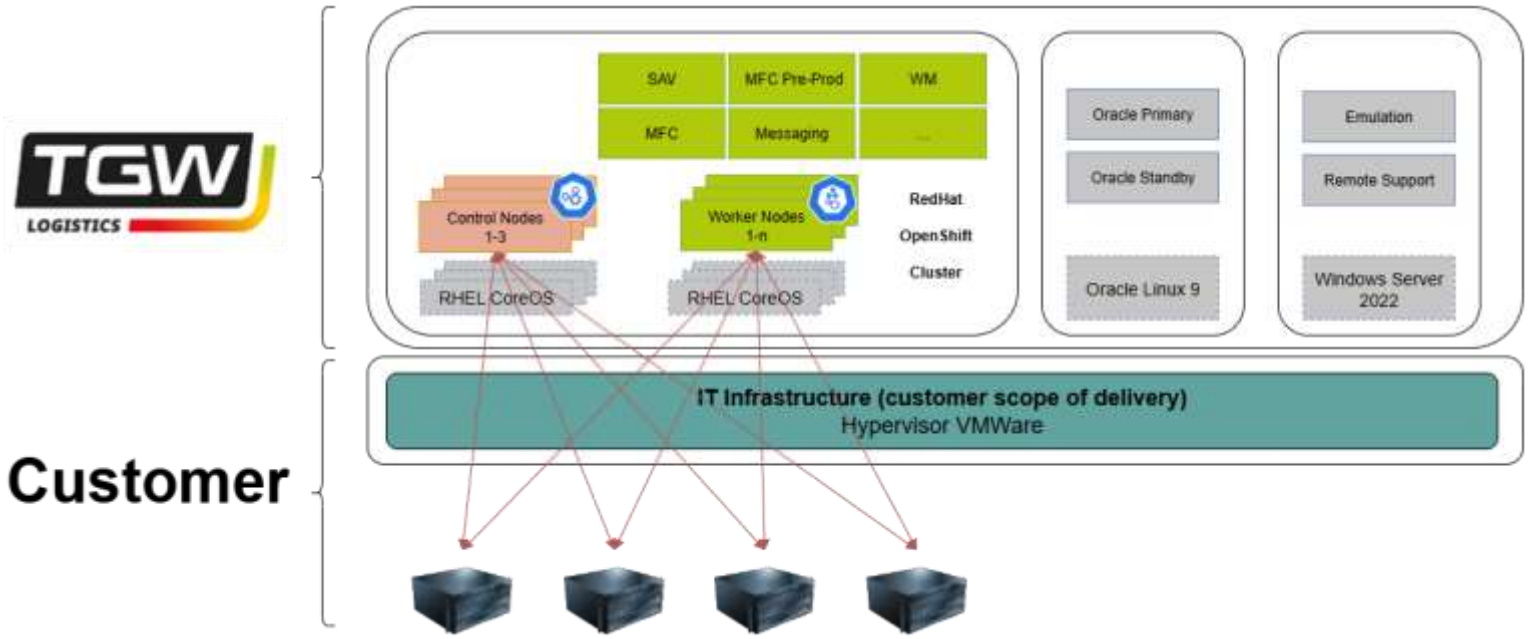


Challenges Storage

In grey box we‘re now running on top of the hypervisor – but need to interact to with storage/nodes below.

But…

Reluctant to give us storage mgmt access for CSI driver, not seeing the need for it altogether (just add storage to the VMs,…), much more complicated from security point of view

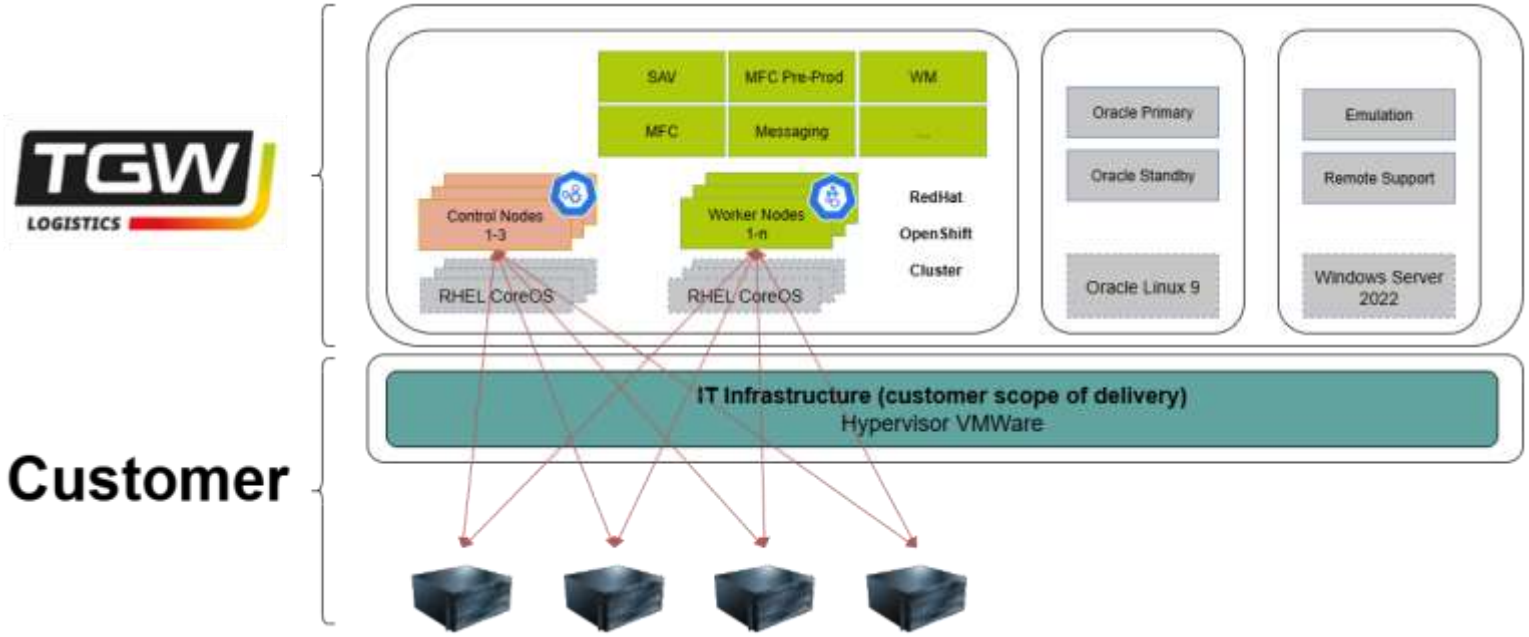


Challenges Storage

We were not addressing storage like rook/ceph (yet).

It definitely brings up lots of questions and uncertainties regarding support.
Shy away of complexity / subscription costs.

Part of it can be solved during pre-sales / contractual phase.

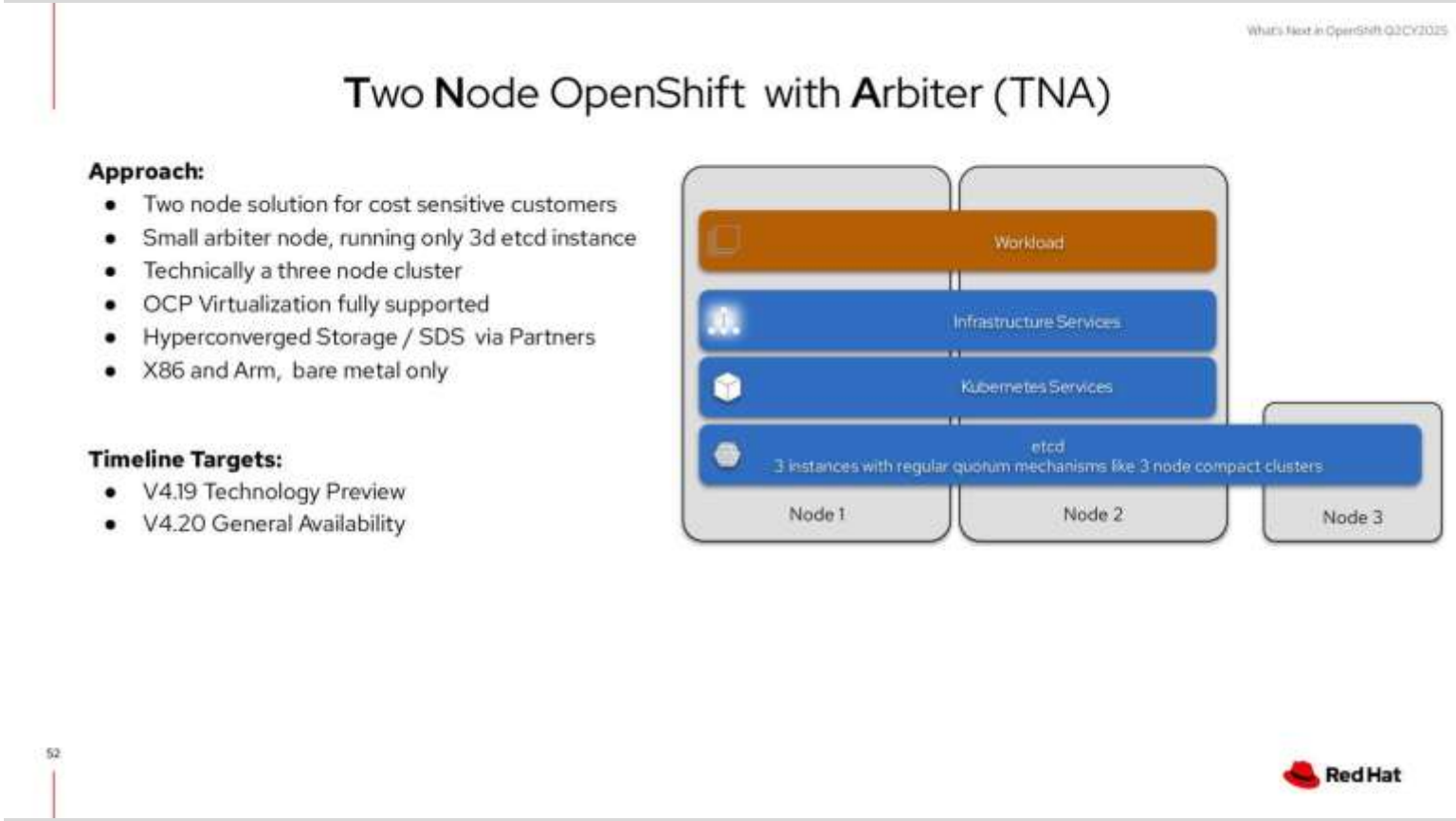


outlook

Outlook
Arbiter Nodes

Many / most customers only provide two fault domains – which is not easy to fit 3 control nodes onto.

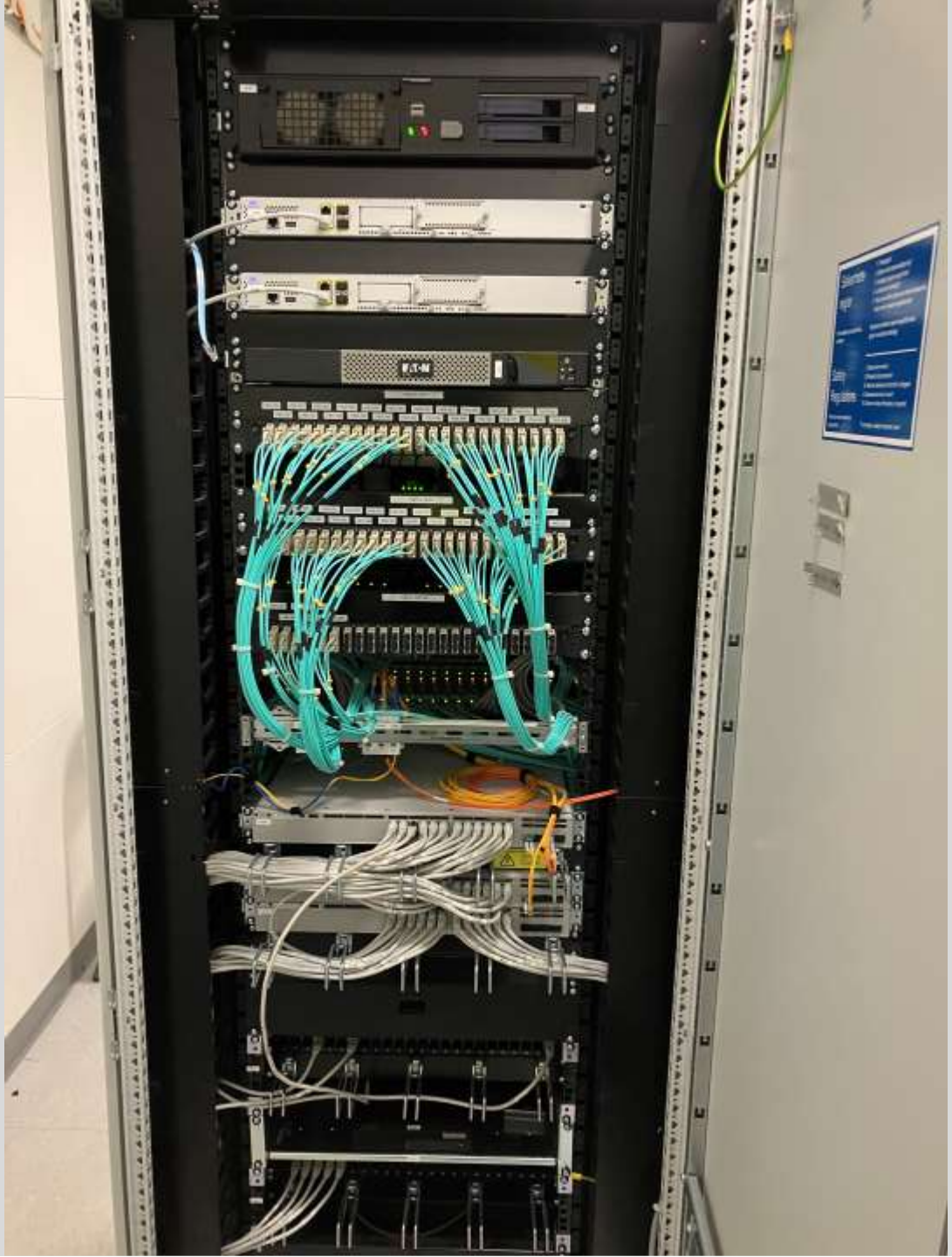
4.19 / 4.20 bring arbiter nodes that we plan to use: f.e. 2 nodes in DC, 1 arbiter node maybe outside of DC in OT Rack.



Outlook Arbiter Nodes

Regardless of black / grey box we always ship OT Racks for device communication.

Although slower connectivity compared to DC (gigabit only) this may be an option for arbiter node.



Thank you