Journey Of Building Our Kubernetes Platform: Successes, Failures, And Valuable Lessons Learned

Maryam Tavakkoli
Senior Cloud Engineer @ RELEX Solutions

KubeCon and Cloud NativeCon, North America, 2023

Hello!

My name is Maryam Tavakkoli. I am a Senior Cloud engineer at RELEX Solutions, specializing in designing and implementing Cloud and Kubernetes infrastructure.

LinkdIn

Medium

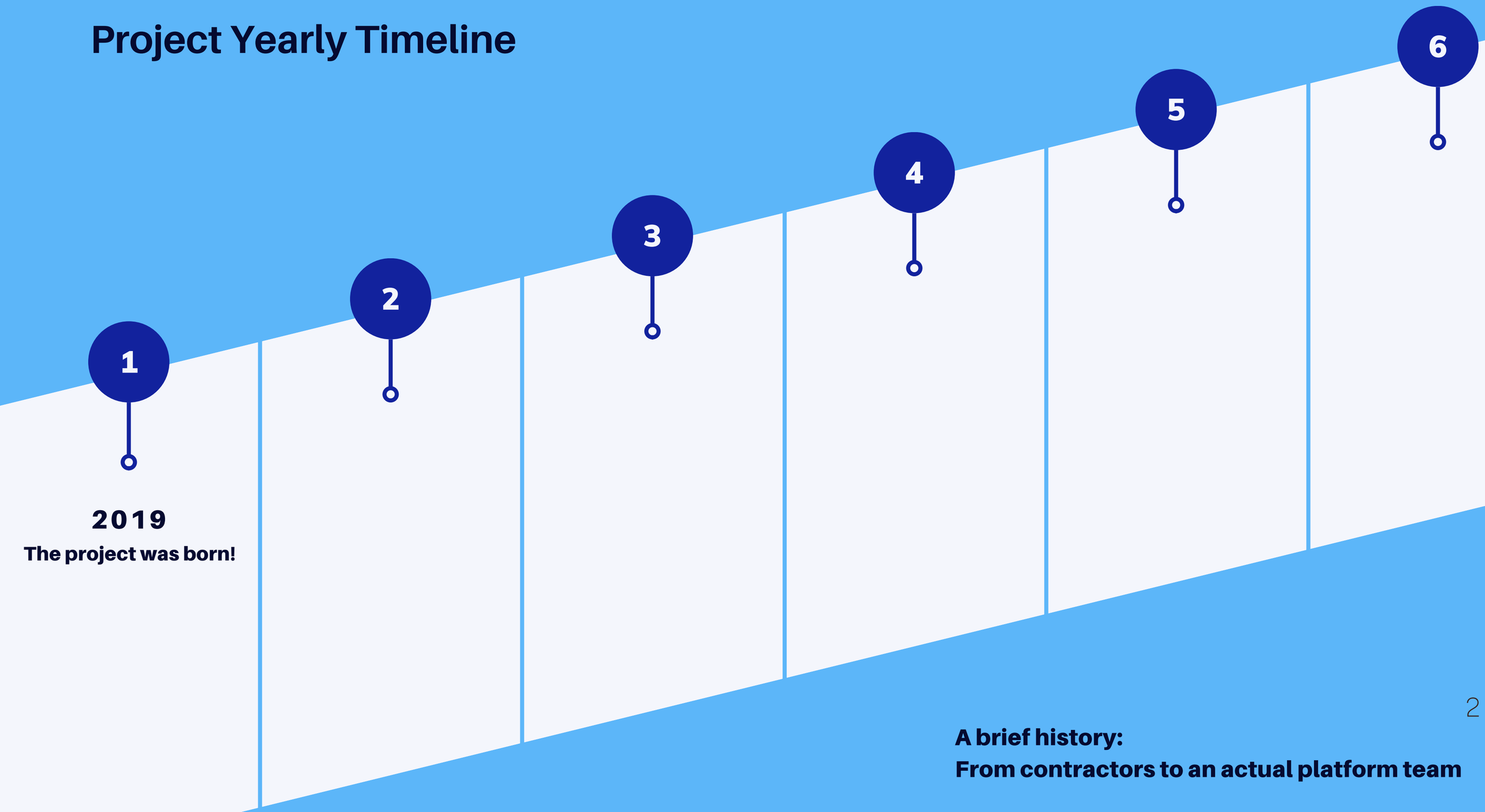
GitHub

Agenda

1. Project timeline: The history
2. Project structure
3. Statistics of Kubernetes Platform usage
4. Advantages of Kubernetes Platform
5. Lessons learned
6. Future roadmap

Project timeline: The history

Project Yearly Timeline



2019

The project was born!

A brief history:
From contractors to an actual platform team

2019 Year Overview

- The project was initially created for a **specific internal development team** that was willing to migrate to Kubernetes at the time.
- As for cloud providers, **Azure** was chosen by management due to other company-related matters (not relevant to the Kubernetes service - AKS)
- A few **consultants** were hired to create the project.
- The project was born at the **end of October 2019.**

Project Yearly Timeline

1

2019

The project was born!

2

2020

The project was changed to be a unified Kubernetes platform within the company.

3

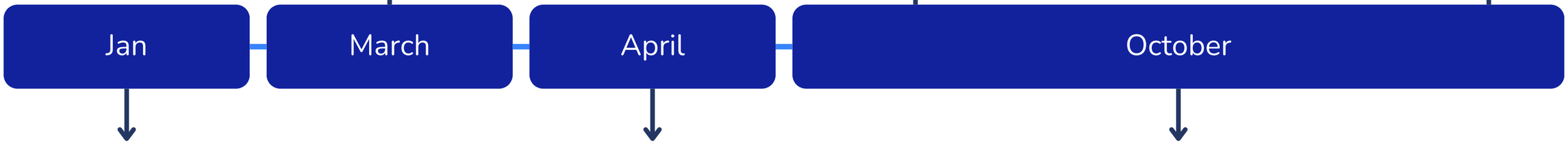
4

5

6

A brief history:
From contractors to an actual platform team

2020 Monthly Progress



I joined the company in January as an **SRE** for one of the development teams, who became a platform user.

By March, there were **5 teams**, which started using the Kubernetes Platform (including my team and I was the responsible person)

In April, there is one **internal employee** starting to work on the project.

In October, a massive MR related to the migration of Service Principles to **Azure Managed Service Identity** (**MSI**) was merged.
It became a major incident and a **revert of the change** was applied within **2 days.**

The **second internal employee** is recruited and joins the team.

One of the development teams decided to **leave** the project as a result!

Failure Scenarios In The Major Incident

What was not working

1

Rights are missing

The Developer had the required permission while testing, but the Users did not have the required access permissions and rights to delete SPs when rolling in.

2

Cert-manager issue

Cert manager did not work properly when Log Analytics was enabled, because LA creates its own Managed Identity and cert-manager can't handle multiple MIs.

3

Ingress-nginx issue

Private Ingress was giving a timeout and did not get an IP address from the internal network due to insufficient SP rights.

4

Authentication

After a clean install, authentication did not work properly (except for the Admin login).

2020 Year Overview

- It was decided to change the scope of the project from a **team-based infra platform** to a **company-wide Kubernetes platform** project.
- There were so many repeated configuration codes among different environments.
- It was decided to use **Terragrunt**, which is a Terraform wrapper:
  - Terragrunt is a thin wrapper for Terraform that provides tools for keeping your code DRY while working with multiple Terraform modules.

Project Yearly Timeline

1

2019

The project was born!

2

2020

The project was changed to be a unified Kubernetes platform within the company.

3

2021

The team grows

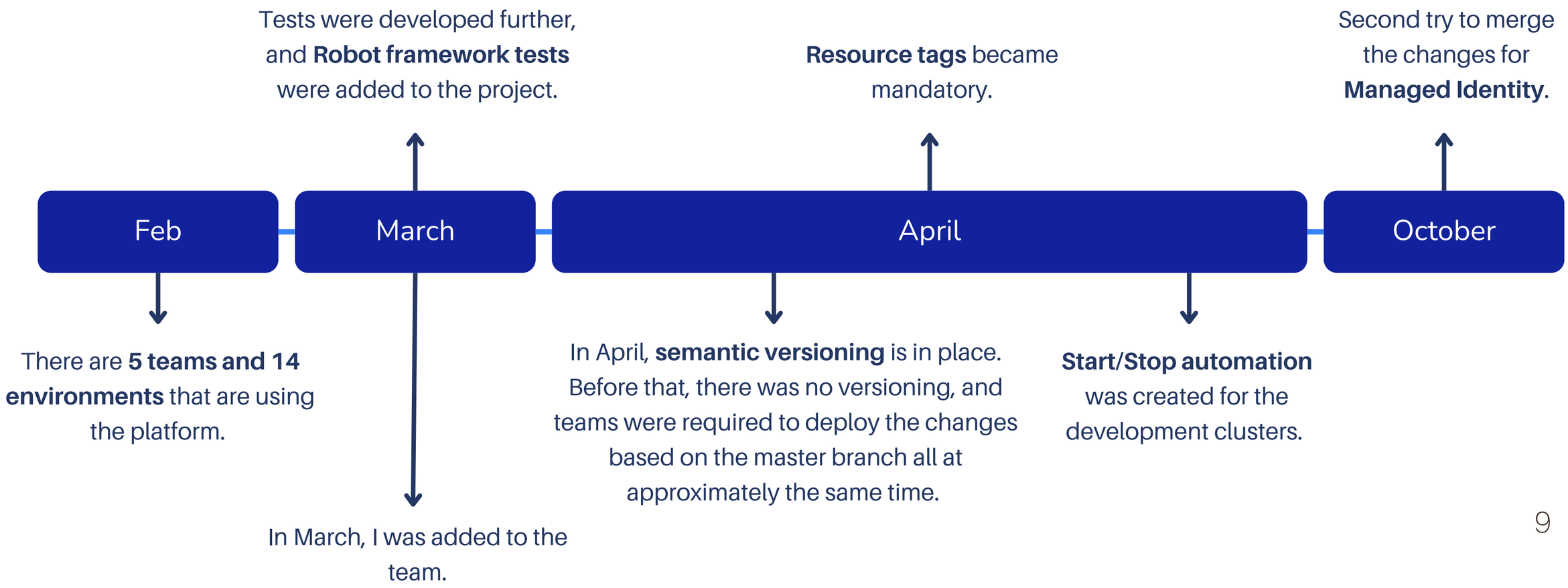
4

5

6

A brief history:
From contractors to an actual platform team

2021 Monthly Progress



2021 Monthly Progress

Full access privileges for platform developers were **dropped**. Instead, **access packages** were created by the Cloud team and access is given to those who activate the package

Security patching is standardized and includes Helm chart upgrades, provider upgrades and terraform/Terragrunt version upgrades.

2021 Year Overview

- As the project grew and more teams were willing to use it, a special team started forming.
- Team members and consultants joined and left until it became a **team of three internal members by the end of 2021.**

Project Timeline

1

2019

The project was born!

2

2020

The project was changed to be a unified Kubernetes platform within the company.

3

2021

The team grows

4

2022

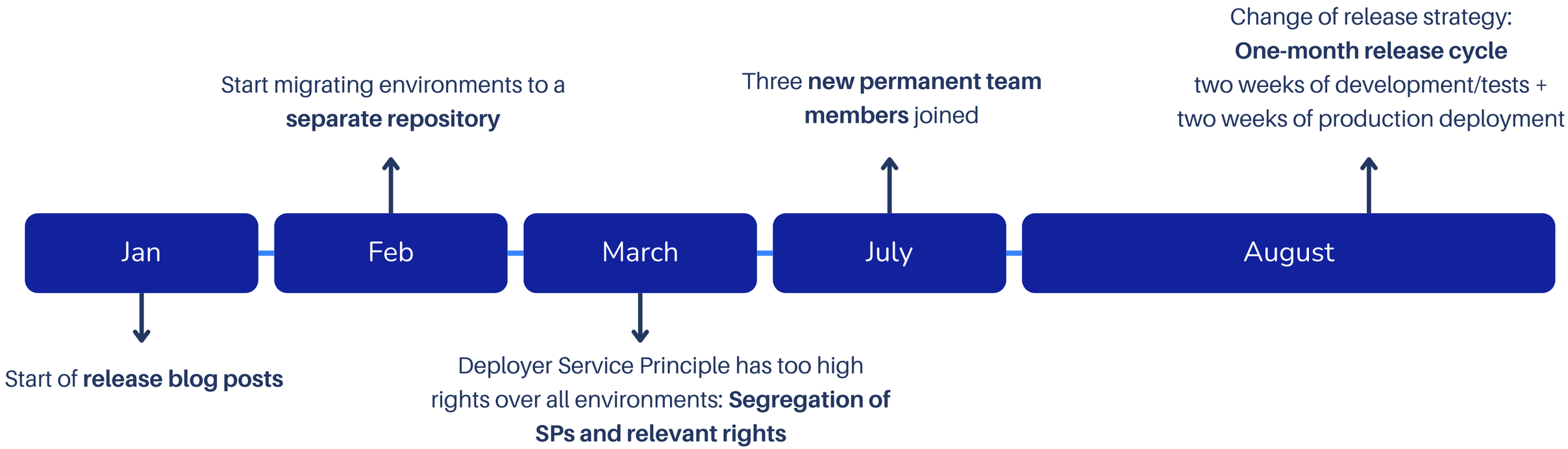
The project gets more mature

5

6

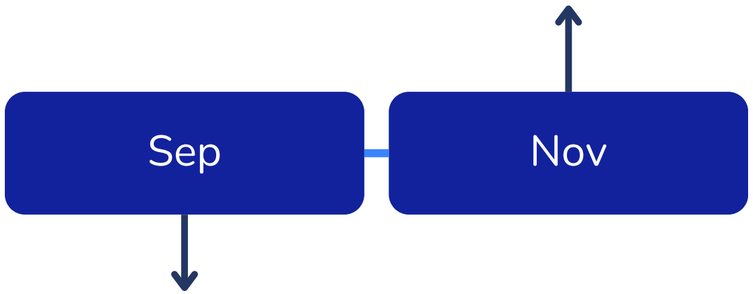
A brief history:
From contractors to an actual platform team

2022 Monthly Progress



2022 Monthly Progress

Start-Stop module is developed to **decrease costs** for dev/test environments



Sort of a Major incident for one of the test environment:
Azure-created RG was deleted manually from the portal by the user:
Add resource locks

2022 Year Overview

- The project became more mature and more people joined the team and it became a **team of six permanent internal members by the end of 2022.**

Project Timeline

1

2019

The project was born!

2

2020

The project was changed to be a unified Kubernetes platform within the company.

3

2021

The team grows

4

2022

The team grows and the project gets more mature

5

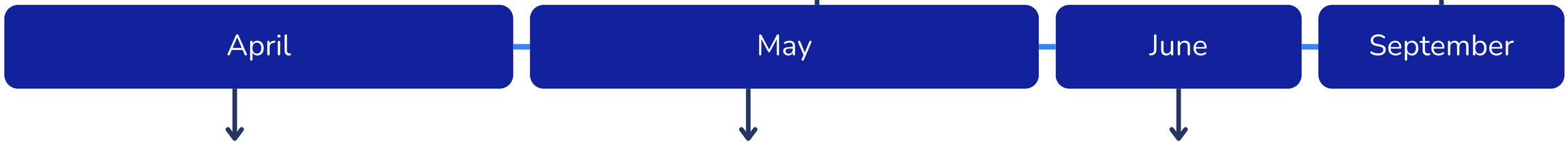
2023

Time for supporting features

6

A brief history:
From contractors to an actual platform team

2023 Monthly Progress



Feature testing process is changed:
Previously:
Continues deployment after each merge to master is tested.
Now:
A temperory test environment creation based on the latest available version and then upgrade to the Master

Fully automated deployments:
Removes the need for teams to run manual steps before the upgrade

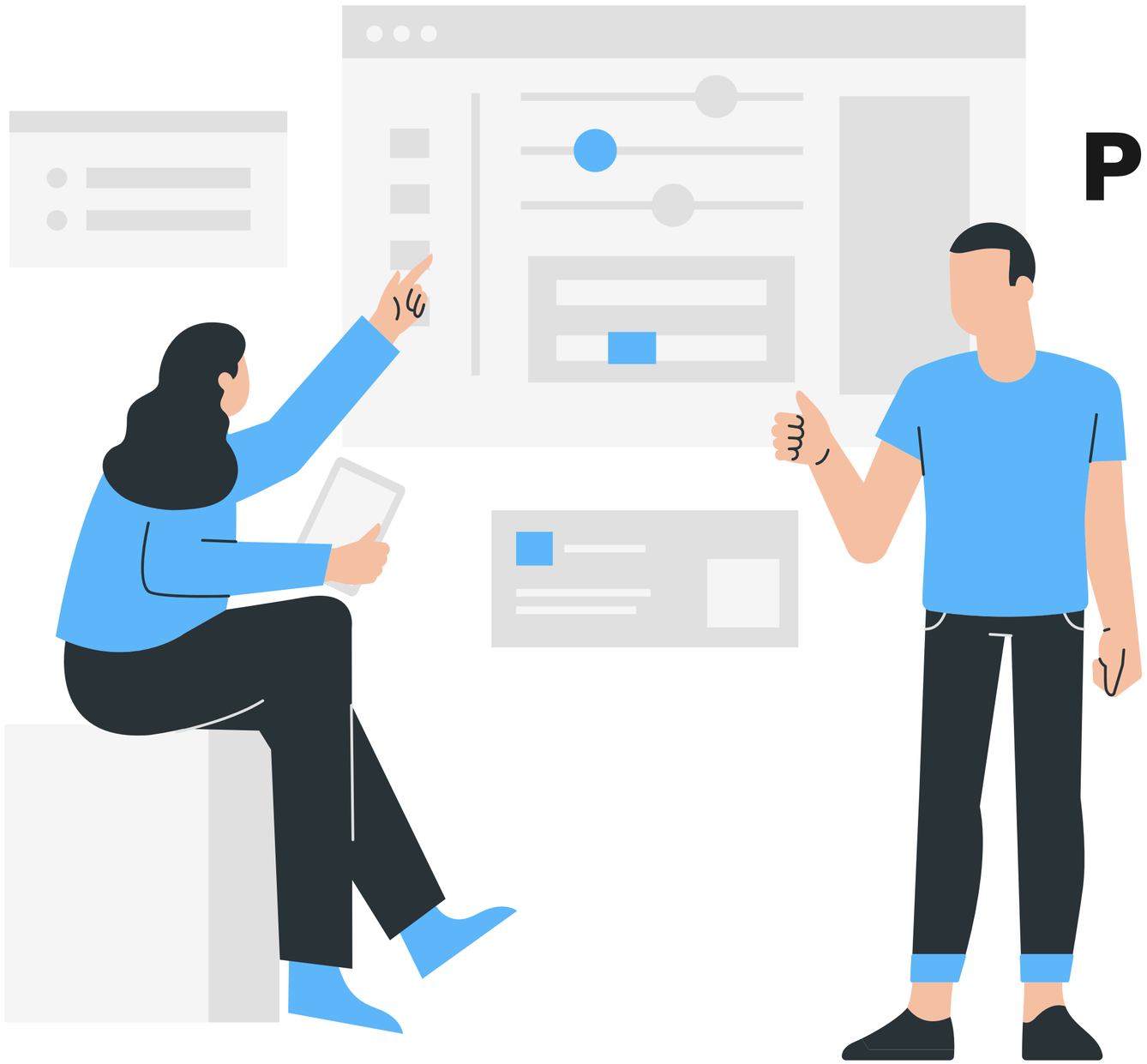
Service Mesh:
Linkerd

Shared cluster:
One cluster for many different teams

Migrating **AKV2K8S** to **External Secret Operator** project

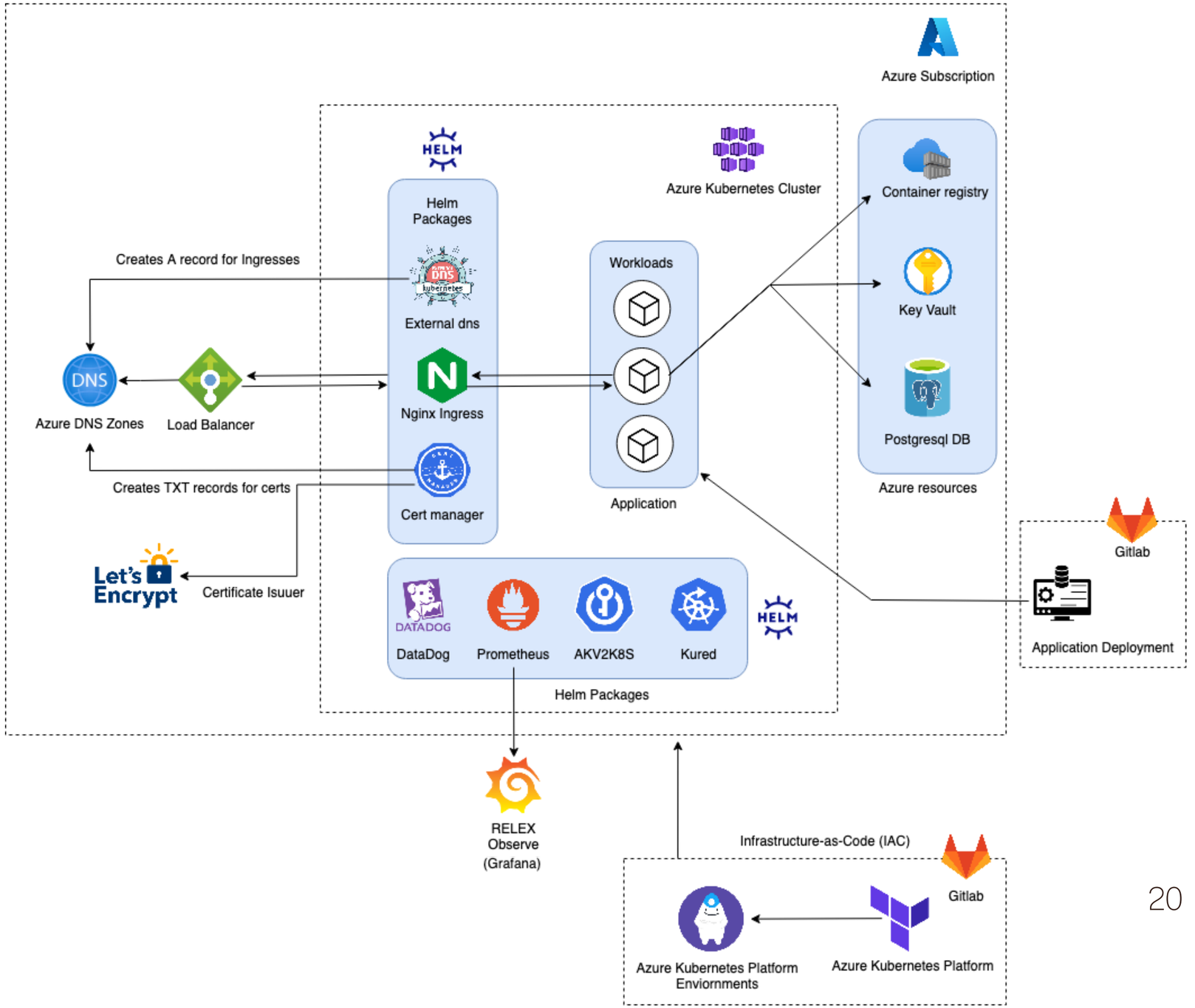
2023 Overview

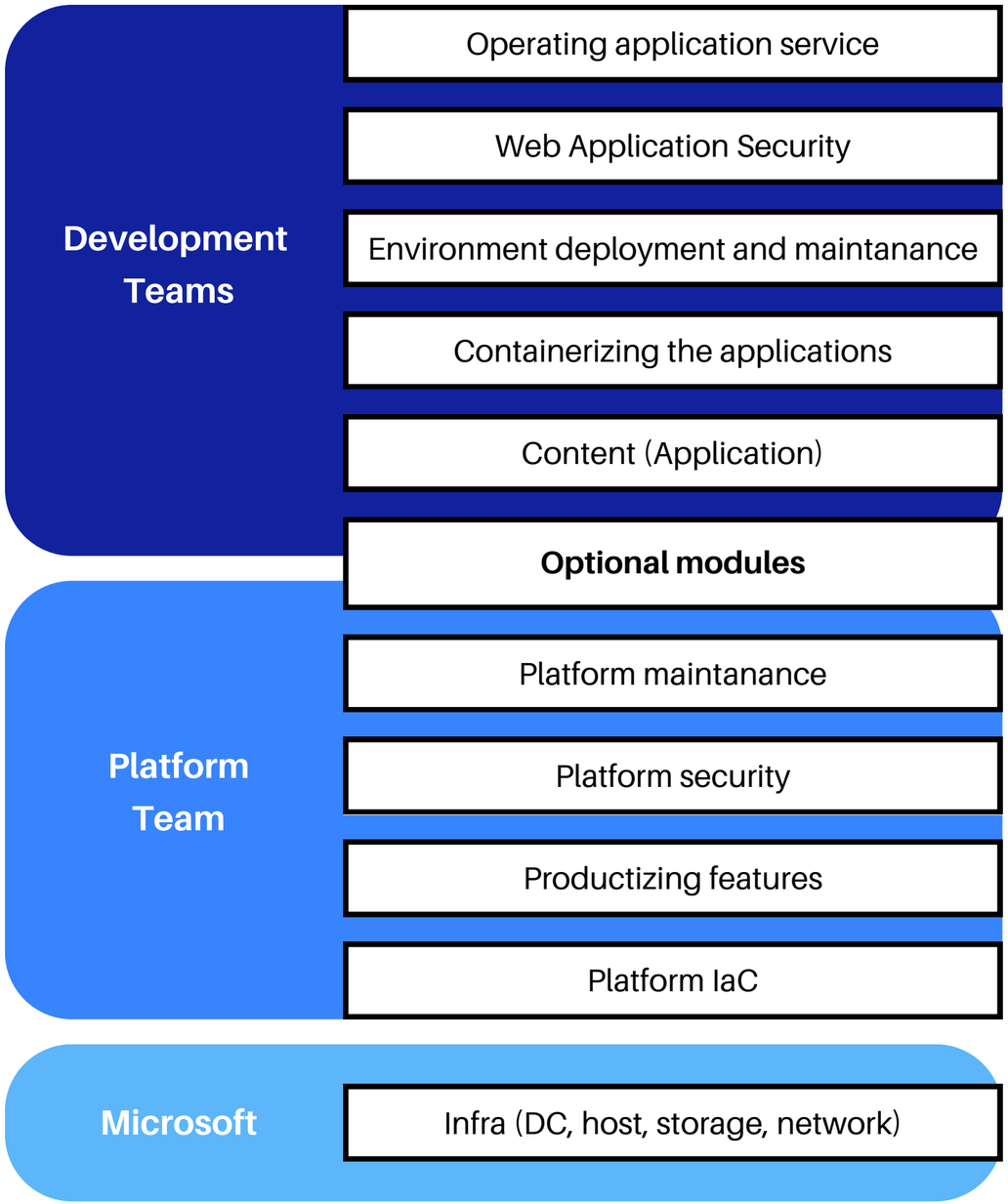
- In 2023, there was enough time to add **supporting features based on team requirements.**



Project Structure

Azure Kube Platform Architecture





Shared Responsibility Model

Development teams, Platform team, and Microsoft

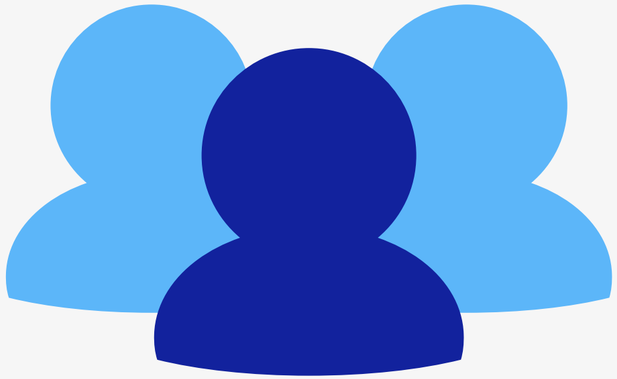
Statistics of Kubernetes Platform Usage

STATISTICS OF KUBERNETES PLATFORM USAGE

22 Mandatory service modules

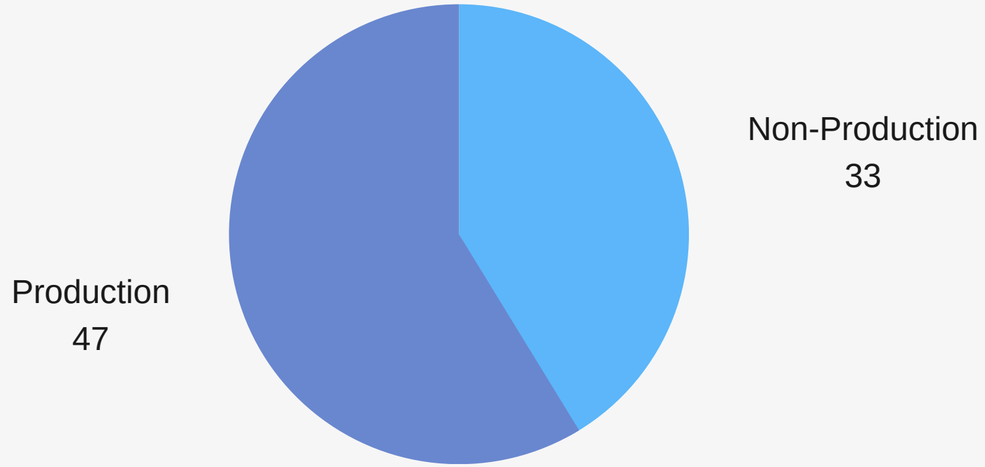
42 Optional service modules

USER TEAMS



13

ENVIRONMENT TYPE



Advantages of Kubernetes Platform

ADVANTAGES OF KUBERNETES PLATFORM

Benefits that it has brought to the company and developer teams

STANDARDIZATION AND UNIFICATION

The project creates **reusable components**, tools, and documentation that make it easier for development teams to work together and follow **consistent practices**.
This results in **improved code quality, reduced development time, and enhanced knowledge sharing.**

SECURITY AND COMPLIANCE

The project **implements the best practices** for access control, encryption, and vulnerability management.

All platform **changes are tested for security** before deployment.

Security patching and upgrades are taken care.

COST OPTIMIZATION

The project takes care of identifying and implementing strategies to make **efficient use of resources.**

This may include rightsizing infrastructure, leveraging cost-effective cloud services, automating resource provisioning and deprovisioning.

FASTER DEVELOPMENT CYCLES

The project can be used to **create and maintain standardized development environments.** This means developers don't have to spend time configuring their local setups. Instead, they can **start coding immediately**, leading to faster development cycles

Lessons Learned

Lessons Learned

What caused the major incident

1

Big Change

- The change was too big:
- There were two different MRs including major changes, which were merged at the same time:
  - Managed identity
  - Cert-manager and Nginx-ingress

2

Not enough testing

- The migration plan was not tested thoroughly.
- There were no proper tests in the project.
- Manual testing scenarios were not extensive enough.

3

No rollback plan

- The plan was overconfident.
- No rollback or failure plan in place.

Lessons Learned

What caused the major incident

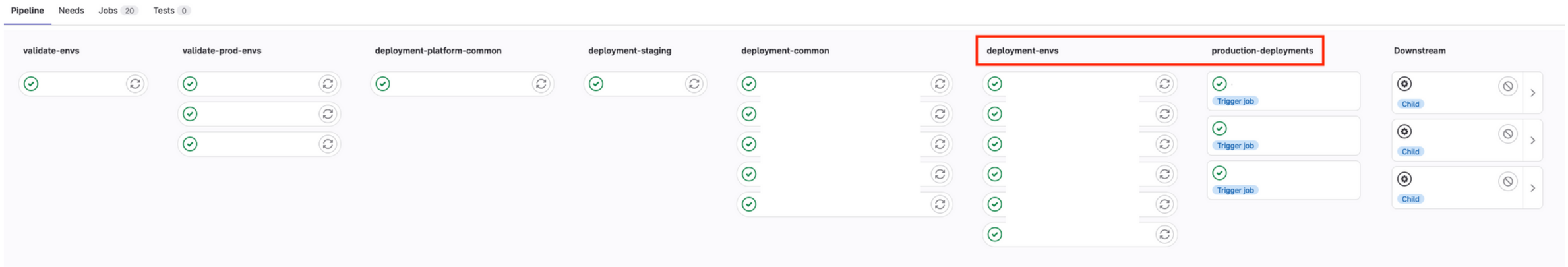
4

Deployment procedure had limitations

5

Major change in immature state of the project

- There was NO versioning in the project
- Deployments from the Master branch
- Dependency of environments to deploy before other changes are merged to the Master
- Automatical deployments to Dev and manual to production
- The whole deployment process should have been more gradual, touching only one instance at a time until fully operational and verified to work.



Lessons Learned

Other aspects

1

Internal team

- Having an internal team (with constant members) is important
- Having consultants is not enough

2

Standardization

- DevOps knowledge is missing in the development teams
- Standardization helps users of such platforms to understand procedures better.

3

Documentation

- Knowledge sharing with development teams
- Best practices blog posts
- Release blog posts with detailed instructions

Lessons Learned

Other aspects

4

Cost optimization

- Think of different possible approaches:
  - Resource tags
  - Stop/Start automation for clusters

5

Feedback

- Development teams are the users, so their feedback matters
- Release cycles should be set considering the availability of the users
- Importance of having a Product Owner

6

Prevent incidents

- Use resource locks when possible
- Set soft-delete policies and backup options

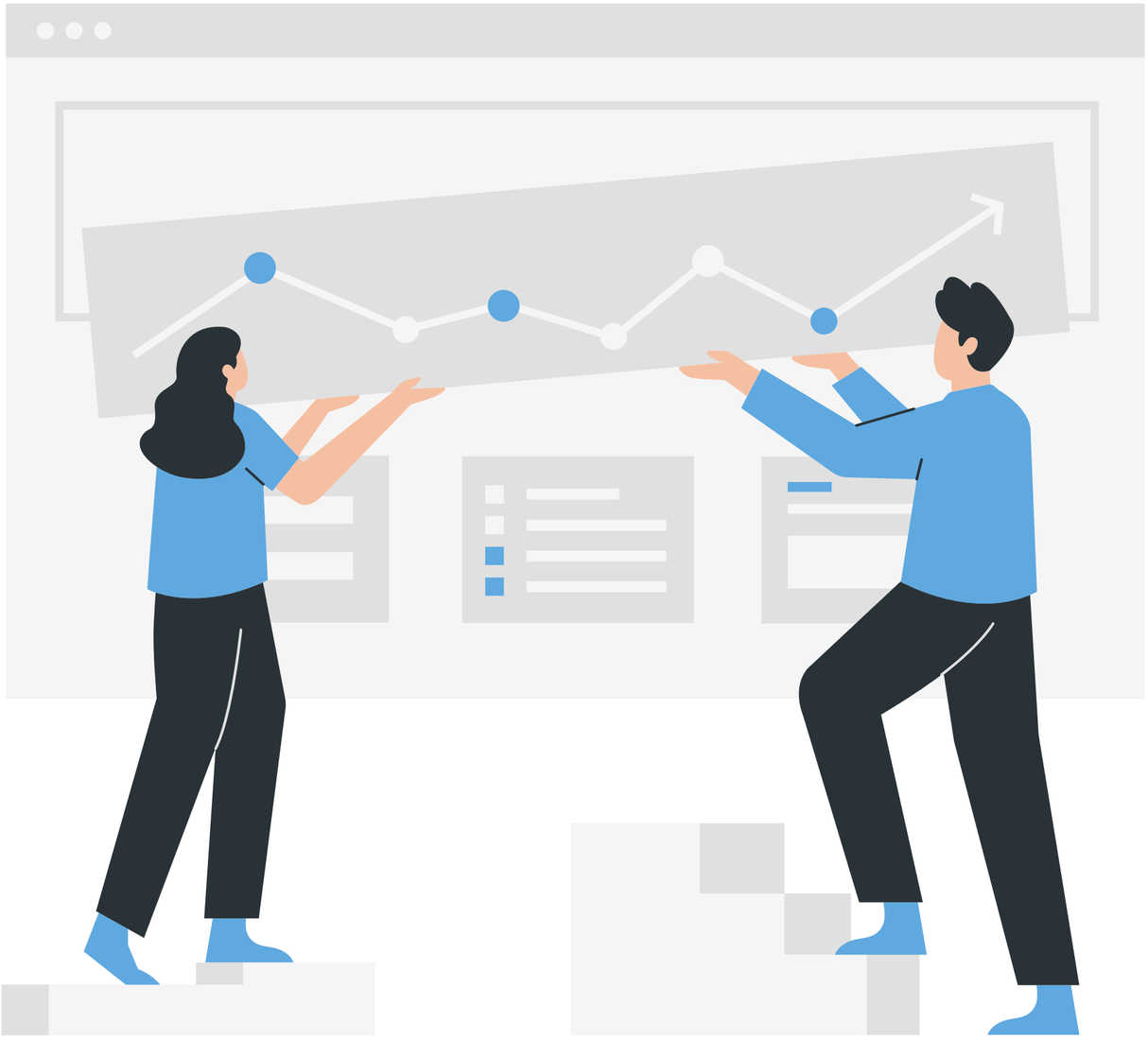
Lessons Learned

Other aspects

7

Security

- Deployer service principles should not have access to every subscription and every resource
- Access packages could be used to segregate access for different users



Future Roadmap

Project Timeline

1

2019

The project was born!

2

2020

The project was changed to be a unified Kubernetes platform within the company.

3

2021

The team grows

4

2022

The team grows and the project gets more mature

5

2023

Time for supporting features

6

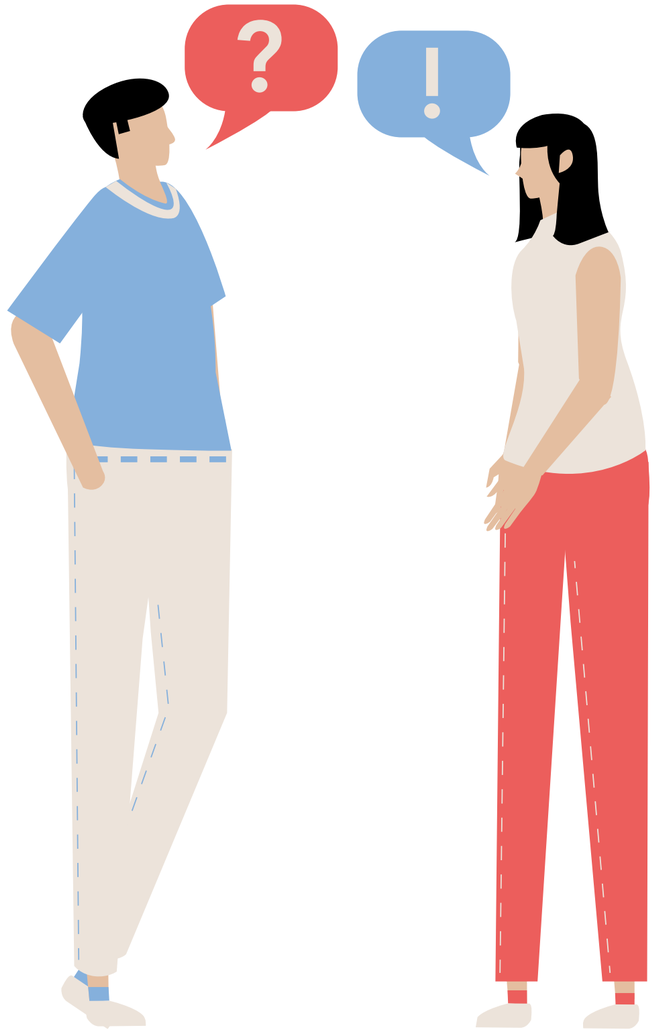
FUTURE

A brief history:
From contractors to an actual platform team

Future Plans

- Service Mesh (LinkerD)
- Shared Cluster (to be used by multiple developer teams)
- Global load balancing for stateful applications
- Web application firewall (WAF)
- Enhancing testing strategies

Questions?



PLEASE SCAN THE QR CODE ABOVE
TO LEAVE FEEDBACK ON THIS SESSION