Comparison between ingress controllers and methods to secure web applications and the kube-API server in kubernetes clusters

Presenter → Dikshita Kalita
Supervisor → Prof. Dr. Martin Leischner
Mentor → Richard Clauß
Date → 13.12.2022

Agenda

01 Research Questions

02 Comparison of ingress controllers

03 Securing web applications

04 Securing the kubernetes API

05 Demo

01

Research Questions

1. **What are the most important differences between the most commonly used ingress controller implementations?**

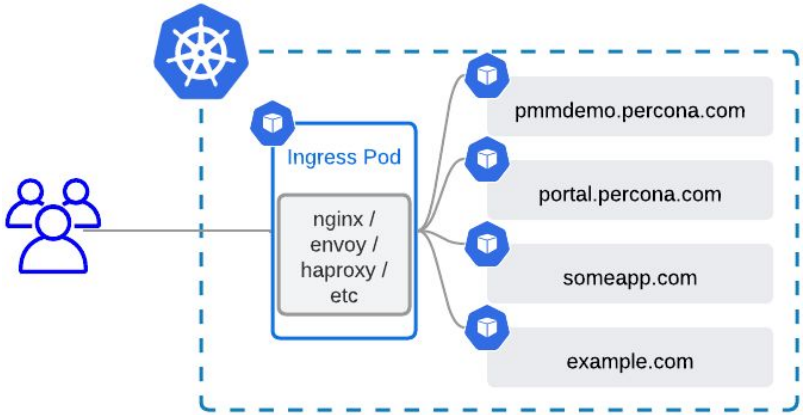
2. **How can web applications running in a kubernetes cluster be secured?**
   (Determine and compare different methods for authentication and authorization)

3. **How can the kubernetes API be secured?**
   (Determine and compare different methods for authentication and authorization)

02

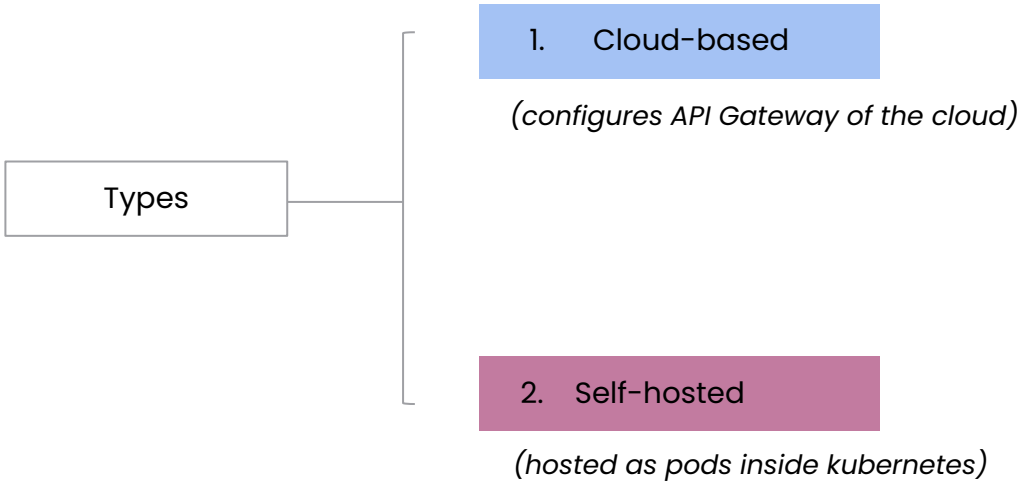
Comparison of ingress controllers

02 Ingress controllers | Introduction



- Operate at the application layer and currently only support HTTP and HTTPS.
- All requests are sent to a single external IP, which is accessible from outside the cluster
- Ingress controllers match URLs to services
- Kubernetes resource IngressClass can define different implementations that can be used in parallel

02 Ingress controllers | Types



02 Ingress controllers | Choosing software

Problem:

Tremendous amount of IngressController implementations
(There are multiple implementations for each web-server)

Approach:

- For each web-server include most commonly used IngressController implementation, which is based on it in the comparison.
- Popularity is determined by amount of github stars

Results (Winners) :

HAProxy → voyager (1.3k ★)

Nginx → ingress-nginx (14k ★)

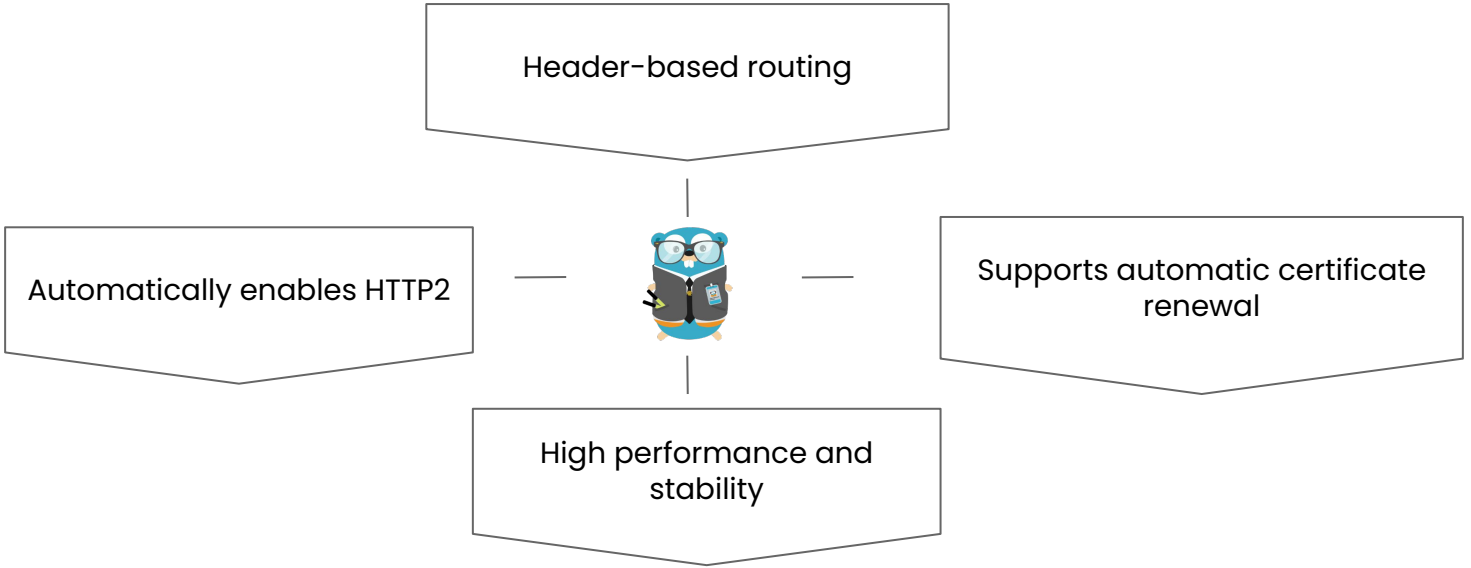
Traefik → Traefik (40.8k ★)

02 Ingress controllers | Comparison

	ingress-nginx	voyager	Traefik
Based on	nginx	haproxy	traefik
Supported protocols	• HTTP 1/2 = tcp • HTTP 3 = udp • TLS	• HTTP 1/2 = tcp • HTTP 3 = udp • TLS	• HTTP 1/2 = tcp • HTTP 3 = udp • TLS
Traffic routing capabilities	• Host • Path • Headers • Method • Query Params	• Host • Path	• Host • Path • Headers • Method • Query Params
Load Balancer Algorithms	• Round robin • Sticky Sessions • Ring Hash • Exponential-Weighted-Moving-Average	• Round robin • Sticky Sessions • Least Connections	• Round Robin • Sticky Sessions • External load balancing

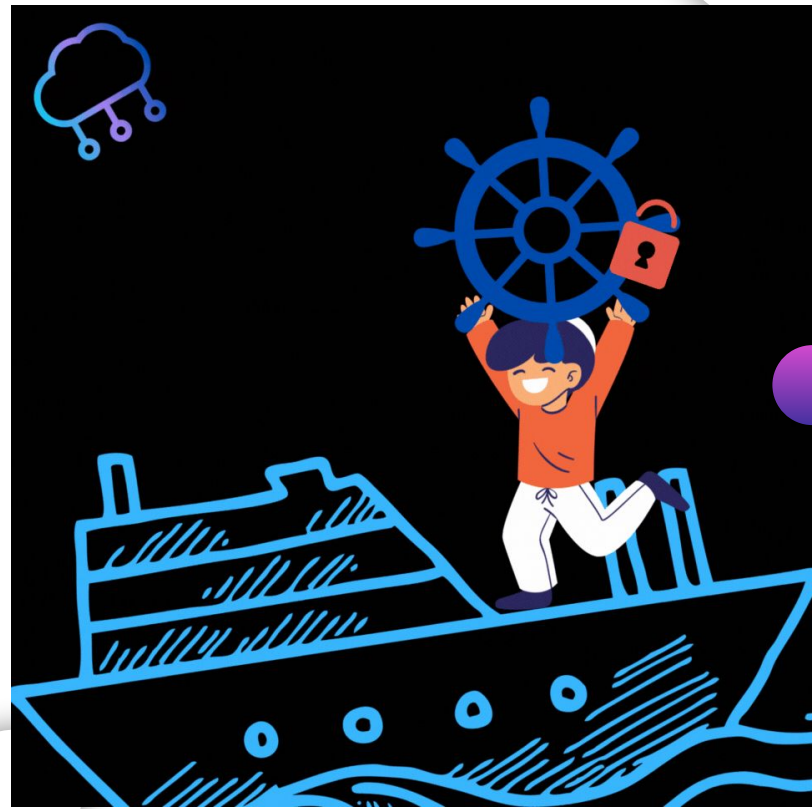
02 Ingress controllers | Preference

IngressController choosen for subsequent work: **Traefik**



03

Securing web applications



03 Securing web-apps | Possibilities

How can an ingress controller handle authentication and authorization if it is deployed as reverse proxy in front of an web-application?

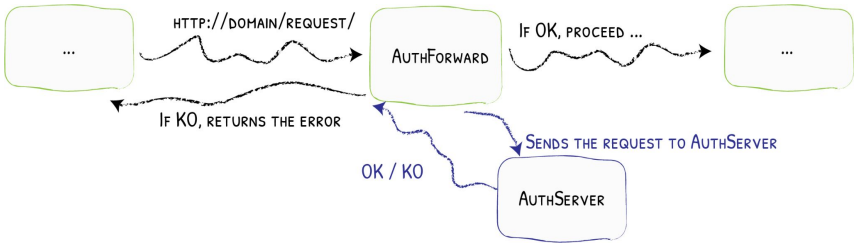
1. Do nothing

→ "Authorization" header and cookies forwarded **without** modification

2. IngressController protects backend by requiring a login

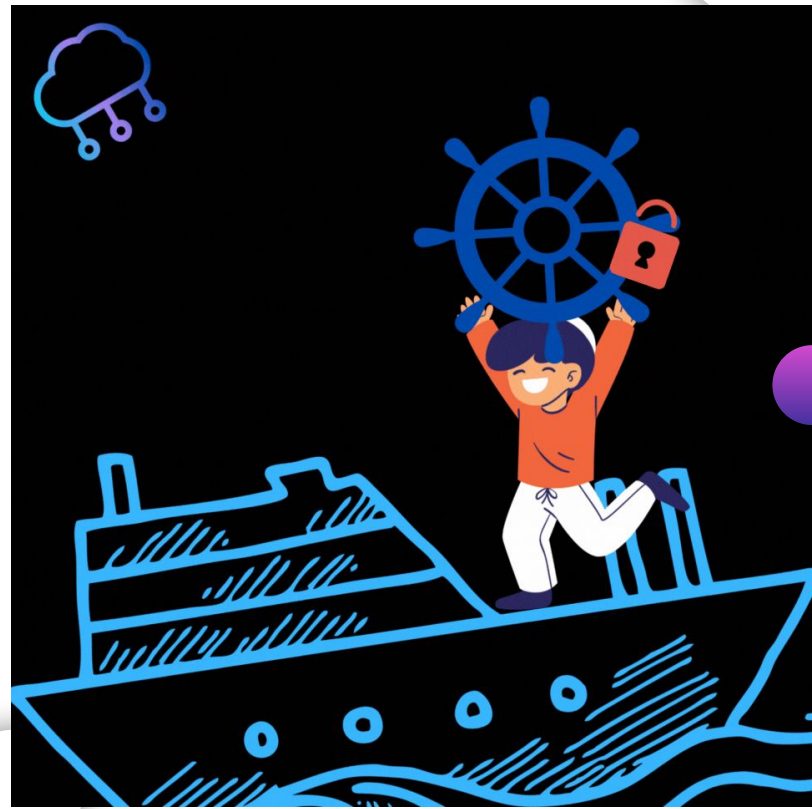
→ For example basic authentication
→ Backend still independent, but may receive ***"Trusted Headers"*** for Single-Sign-On

3. IngressController uses external authentication server



04

Securing Kubernetes



04 Securing Kubernetes | Introduction

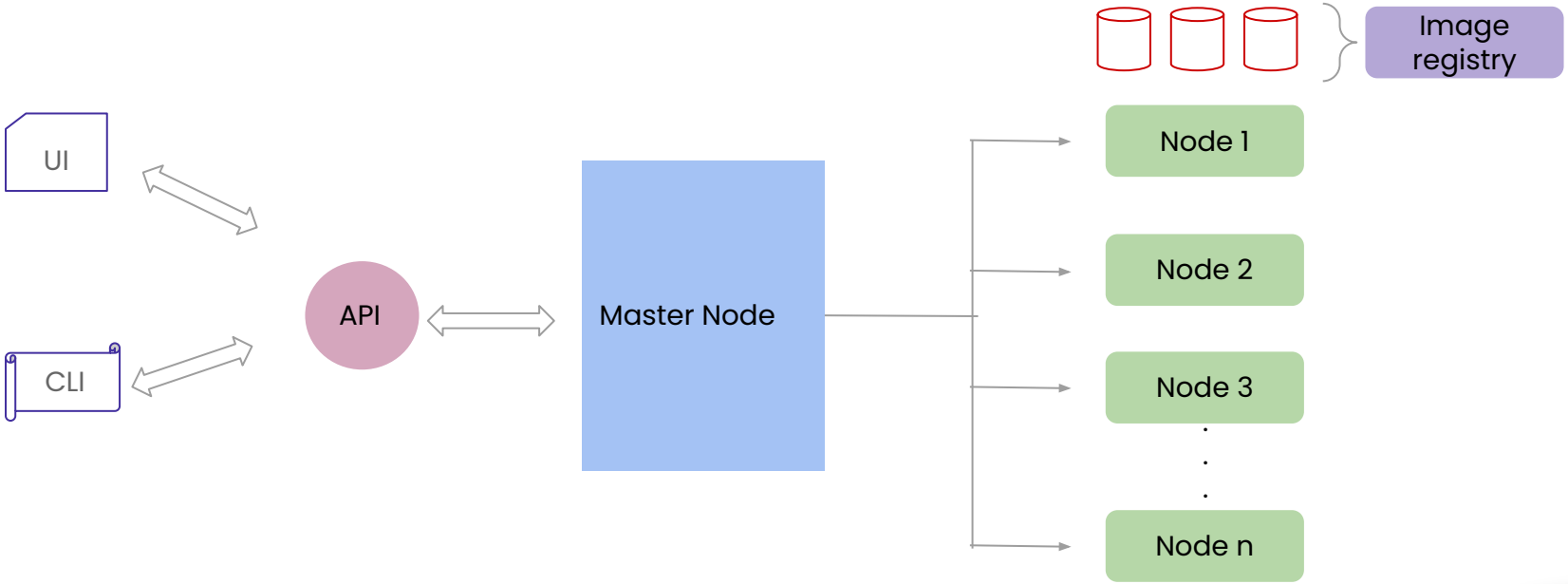
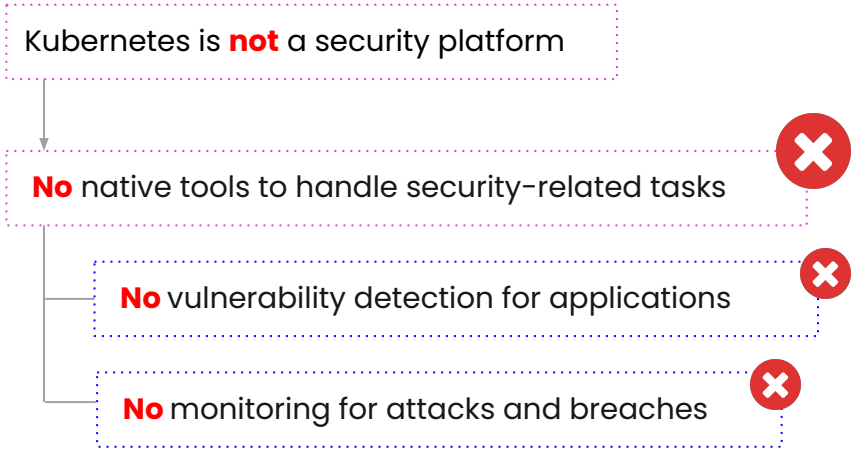


Fig: Architecture in bigger picture

04 Security in Kubernetes | Access Control



Solution:

Kubernetes Security Pipeline

Consists of modular and exchangeable parts

Main component to protect:

Kube API

04 Security in Kubernetes | Security Pipeline

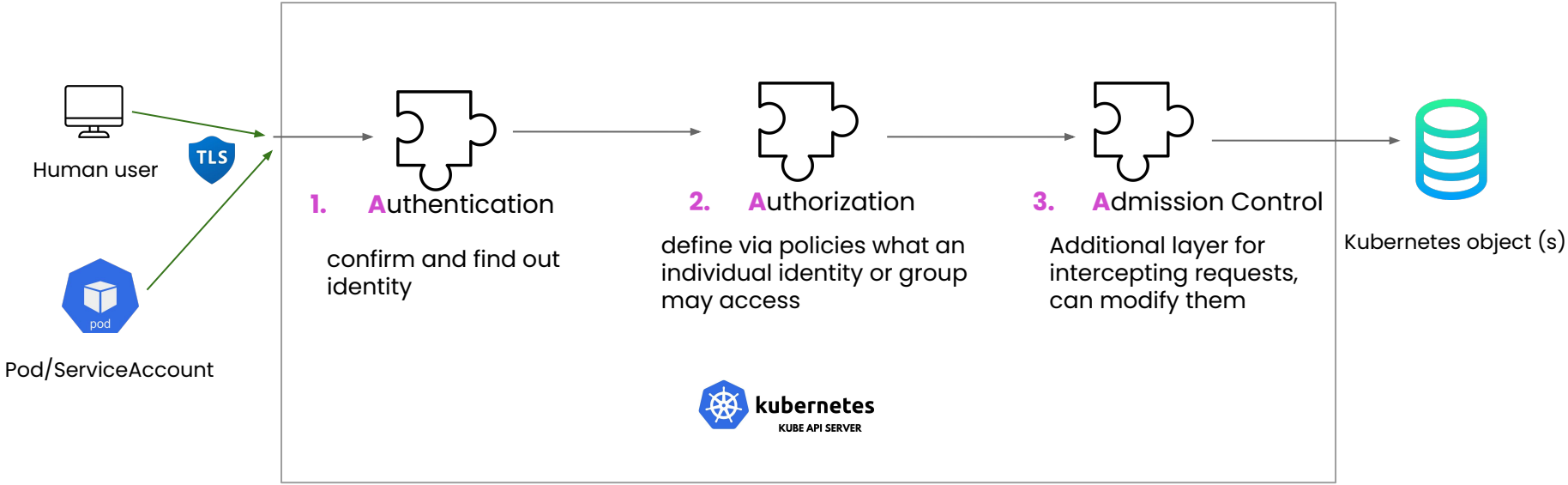


Fig: Kubernetes security pipeline, Source: Click here

04 Security in Kubernetes | Authentication

Static Token File

- Pass a CSV with the following:

<token>,<username>,<UID>,"<group1, group2>"
<token>,<username>,<UID>,"<group2, group4>"

(HTTP Bearer authentication)

- Not scalable

X509 Client Certificates

```
root@masterdev:/var/lib/rancher/k3s/server/tls# cat client-ca.crt
-----BEGIN CERTIFICATE-----
MIIBdzCCAR2gAwIBAgIBADAKBggqhkjOPQQDAjAjMSEwHwYDVQQDDBhrM3MtY2xp
ZW50LWNhQDE2Njk2NDU3MjEwHhcNMjIxMTI4MTQyODQxWhcNMzIxMTI1MTQyODQx
WjAjMSEwHwYDVQQDDBhrM3MtY2xpZW50LWNhQDE2Njk2NDU3MjEwWTATBgcqhkjO
PQIBBggqhkjOPQMBBwNCAASBll28qpxgEkZlrKH/ZWcu9hxv8PSVlrC1TK1I/CsK
g8GaDkoUl7r7M3dmM87Mh5h/9wRkIxd0JvgT7EN7zCCdo0IwQDAOBgNVHQ8BAf8E
BAMCAqQwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4EFgQUmfJYGHagEFd7W9P5bX8/
A5l9F2gwCgYIKoZIzj0EAwIDSAAwRQIgDzZi/E/MlPAwD+DQzU+f7ryqu+Tv15kZ
5/fKrL7nfLYCIQD1cJgFyyJ3F4lEEYQyt7cNEg5M8V6mBizNAZu1NsQXeA==
-----END CERTIFICATE-----
```

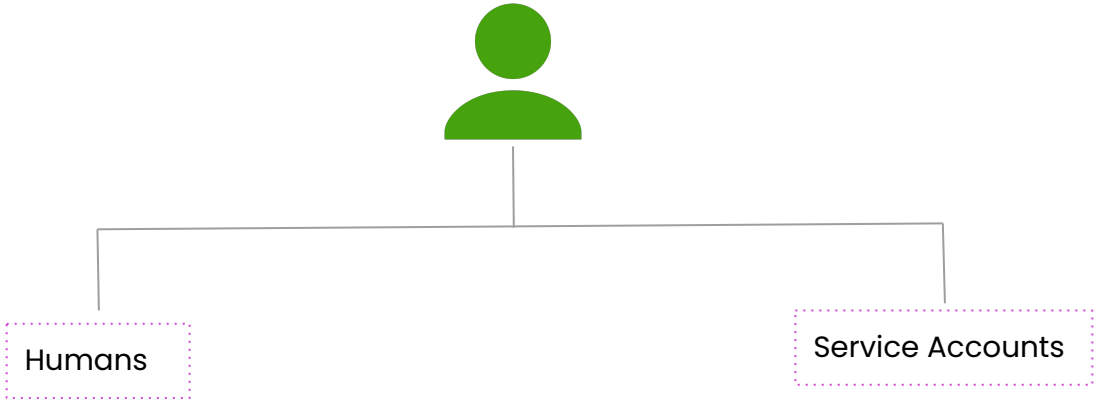
- Insecure
- Long-lived and can't be revoked effectively
- Makes it hard to use groups with RBAC

Dynamically managed Tokens

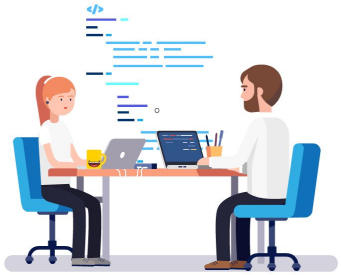
- OpenID Connect (OIDC)

(HTTP Bearer authentication)

04 Security in Kubernetes | Categories of users

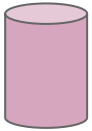


04 Security in Kubernetes | Human users



For human users

Kubernetes does not have



User, profile database / lookup table to store usernames, passwords to store usernames, passwords

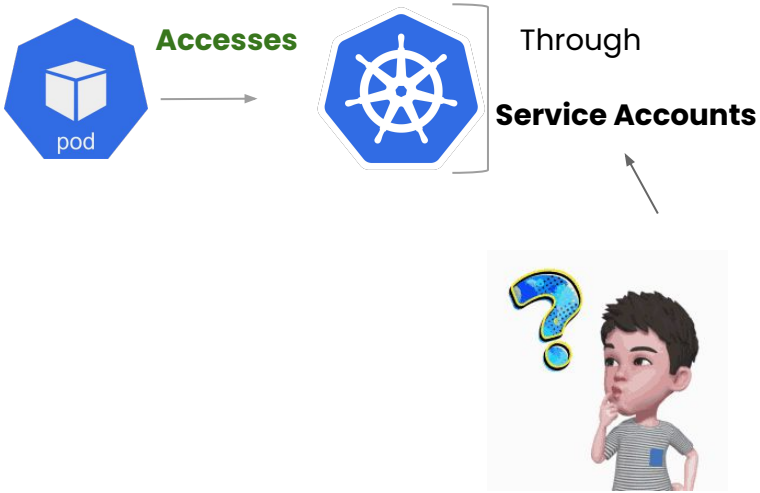
i.e.

NO API calls to create user

Instead

RELIES on a **variety of techniques** to delegate that

04 Security in Kubernetes | Service Accounts



Service Account / SA is

1. Internal representation of a set of credentials that are typically stored as Secrets.

2. Pods uses this SA to authenticate when they talk to internal API endpoint

3. By default, SA have no access permissions but can be configured using RBAC to give them access permission and role-based control so they can query and manipulate

04 Security in Kubernetes | Service Accounts

Very useful if we have an automatized application set-up in the cluster to deploy the pods after its build up

1.
```
root@masterdev:/home/masterdev# k3s kubectl create serviceaccount sa -n default
serviceaccount/sa created
```

2. Create a Role:

   k3s kubectl apply -f default-role.yaml

3. Create a RoleBinding:

   k3s kubectl apply -f role-binding.yaml

04 Security in Kubernetes | Authorization

Authorization modules:

1. AlwaysAllow
2. AlwaysDeny

Not for production **but for** specified use-case

3. Node — Using this node or kubelet would talk to API server
4. Attribute-Based Access Control (ABAC)
5. Role-based Access Control (RBAC)

1. It evaluates attributes or characteristics in order to determine the access

2. The administrator defines a set of user authorization policies into a file with one JSON per line format.

Any modification in the file requires API server to be restarted.

1. Here we define **Roles** and with these roles a user can access kubernetes objects

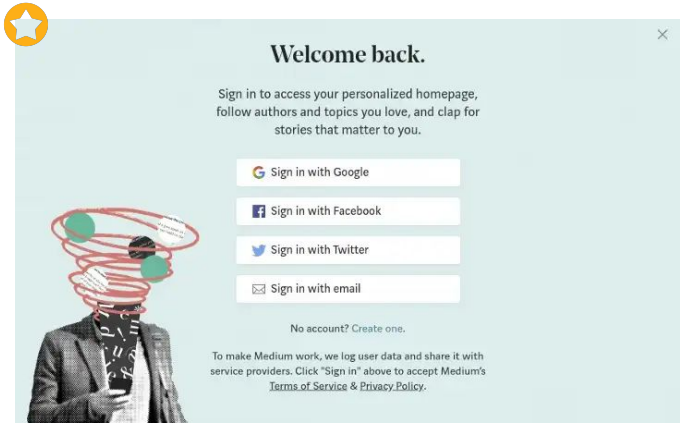
6. Webhook — We can connect multiple webhooks for authorization

04 Security in Kubernetes | Auth | OIDC

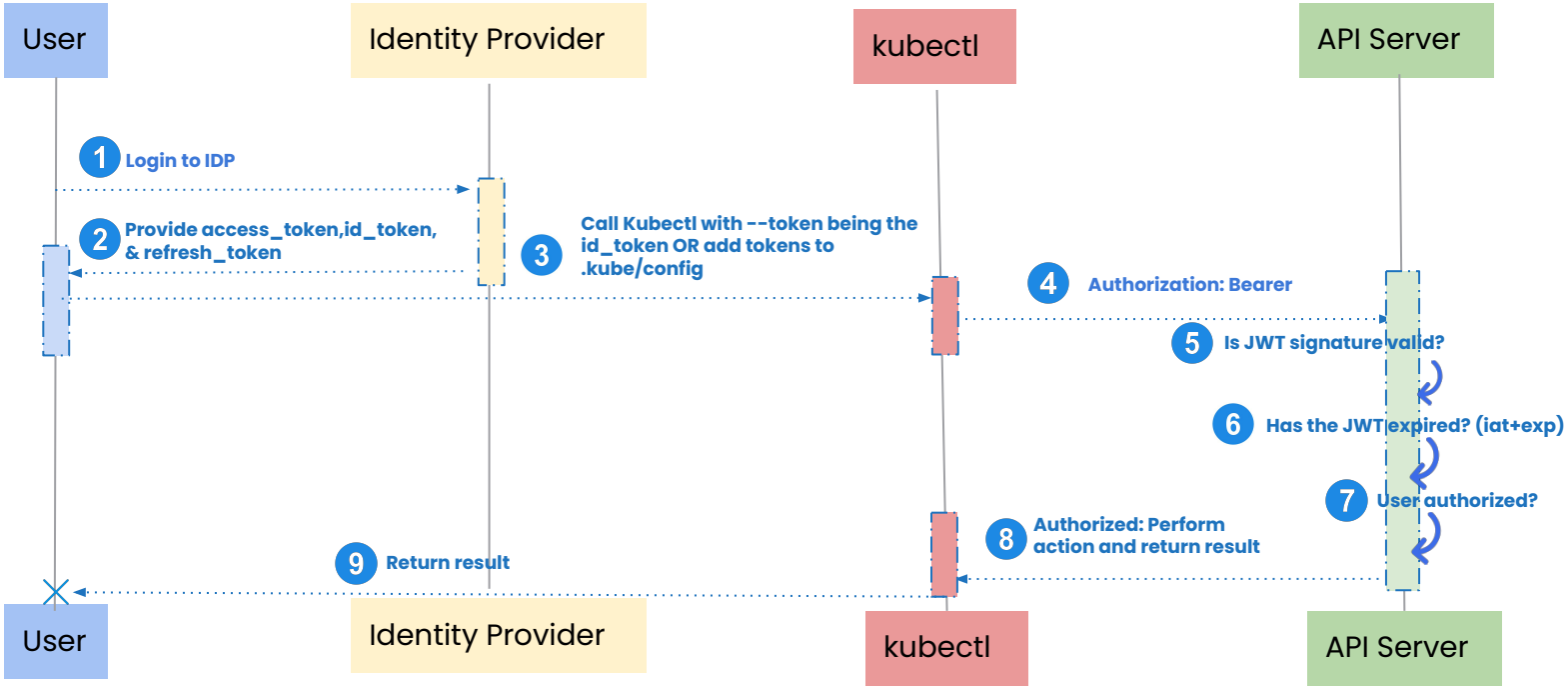
1. OpenID Connect or OIDC is an identity protocol that can be used for **authorization and authentication**
2. It aims to be easy to use and extends the existing technology OAuth2.0
3. Typical use-case: Identity Federation

Simple identity layer for **Authentication**

Standard protocol for **Authorization**



04 Security in Kubernetes | Auth | OIDC



04 Security in Kubernetes | OIDC | IdP

- A system entity that creates, maintains, and manages identity information for users and also provides authentication services to relying applications.

Does Kubernetes provide an OpenID Connect Identity Provider ?

Solution ?

04 Security in Kubernetes | OIDC | IdP

We can

1. Use an existing public OpenID Connect IdP

- Google
- Microsoft
- Amazon
- Okta
- PayPal
- SalesForce
- PhantAuth
- Github

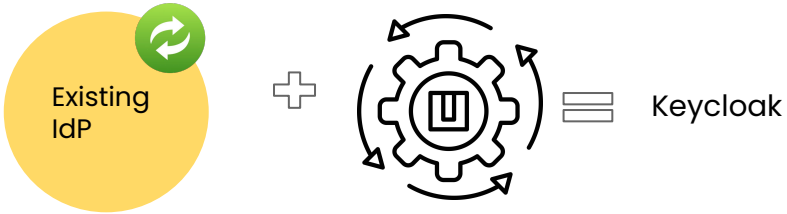
2. Run our own Identity Provider

- Keycloak
- Dex
- CloudFoundry User Account and Authentication
- Tremolo Security's OpenUnison

04 Security in Kubernetes | OIDC | IdP

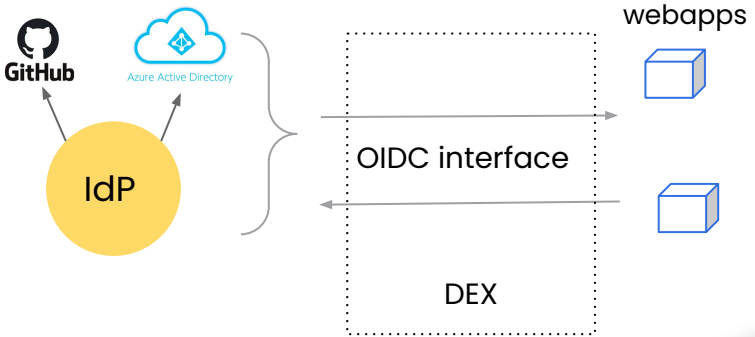
Keycloak

- Complex to operate since it needs standalone database
- **Scenario:**

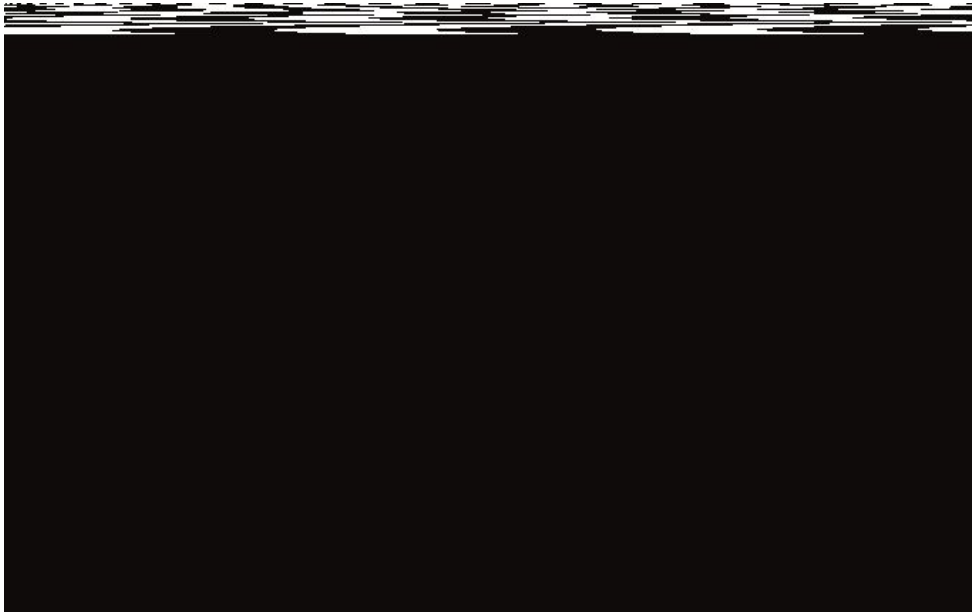


Dex

- Simple and easy to set up
- **Scenario:**



04 Security in Kubernetes | Demo



Sources (1/2)

01. Kubernetes (Publisher). (no date). Accessed on 13. December 2022
from https://kubernetes.io/docs/reference/access-authn-authz/authentication
02. TremoloSecurity (Publisher). (no date). Accessed on 13. December 2022
from https://www.tremolosecurity.com/post/kubernetes-dont-use-certificates-for-authentication
03. Hitesh Jethva. (04.04.2022). Accessed on 13. December 2022
from https://cloudinfrastructureservices.co.uk/oauth2-vs-openid-whats-the-difference/
04. Sysdig (Publisher). (no date). Accessed on 13. December 2022
from https://sysdig.com/learn-cloud-native/kubernetes-security/kubernetes-rbac/
05. Geeksforgeeks (Publisher). (11.07.2022). Accessed on 13. December 2022
from https://www.geeksforgeeks.org/types-of-authentication-protocols/
06. Schuyler Brown. (23.09.2022). Accessed on 13. December 2022
from https://www.strongdm.com/blog/oidc-vs-saml#:~:text=OIDC%3A%20What's%20the%20Difference%3F,to%20obtain%20the%20security%20token
07. Rob Sobers. (05.04.2012). Accessed on 13. December 2022
from https://www.varonis.com/blog/what-is-oauth
08. Dinika Senarath. (28.09.2018). Accessed on 13. December 2022
from https://dinika-15.medium.com/identity-federation-a-brief-introduction-f2f823f8795a
09. Wikipedia (Publisher). (no date). Accessed on 13. December 2022
from https://en.wikipedia.org/wiki/Identity_provider
10. metal-k8s (Publisher). (no date). Accessed on 13. December 2022
from https://metal-k8s.readthedocs.io/en/latest/developer/architecture/authentication.html
11. Scott Chang. (28.06.2020). Accessed on 13. December 2022
From https://medium.com/@sct10876/keycloak-vs-dex-71f7fab29919

Sources (2/2)

12. Traefik (Publisher). (no date). Accessed on 13. December 2022 from https://doc.traefik.io/traefik/v2.0/middlewares/forwardauth/

Note of gesture

Thank you for your attention
and patience