You Shall Not Password

Modern Authentication for Web Services

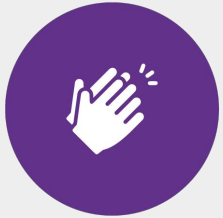
Eli Holderness
they / them / theirs

@eli@hachyderm.io

Code of Conduct

Be aware of others

Be friendly and patient



Be welcoming and respectful

Be open to all questions and viewpoints

Be understanding of differences

Be kind and considerate to others

Thank you to our Sponsors!

You Shall Not Password

Modern Authentication for Web Services

Eli Holderness
they / them / theirs

@eli@hachyderm.io

What authentication is

Common ways to authenticate users

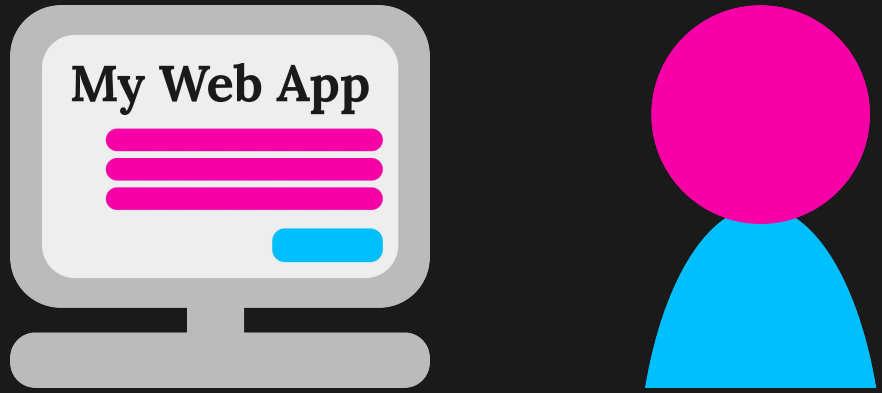
What makes an authentication system good

@eli@hachyderm.io

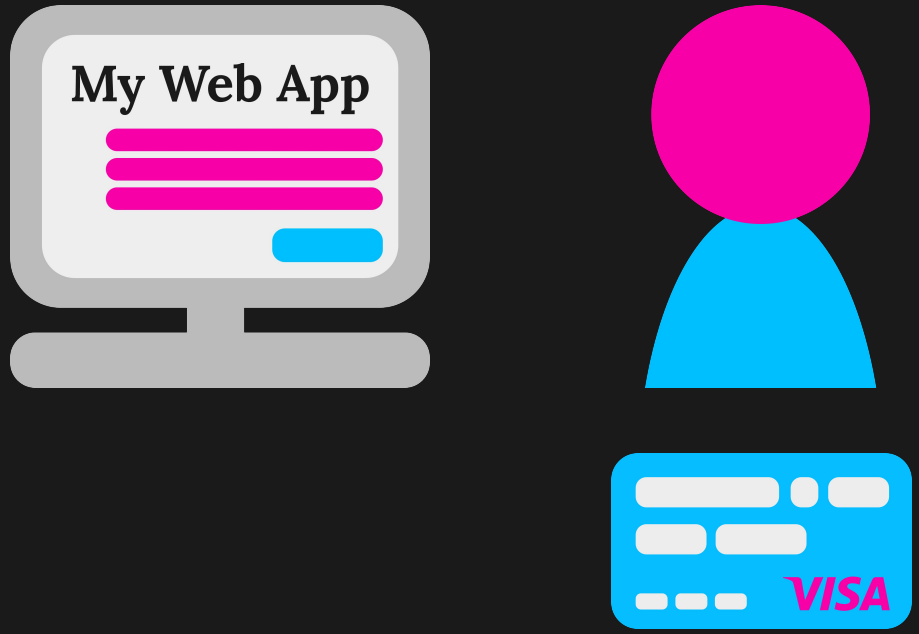
What is authentication?

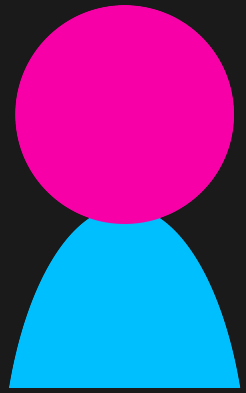
@eli@hachyderm.io

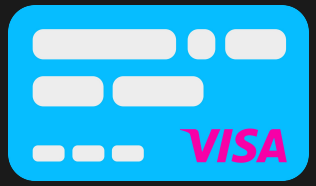
@eli@hachyderm.io



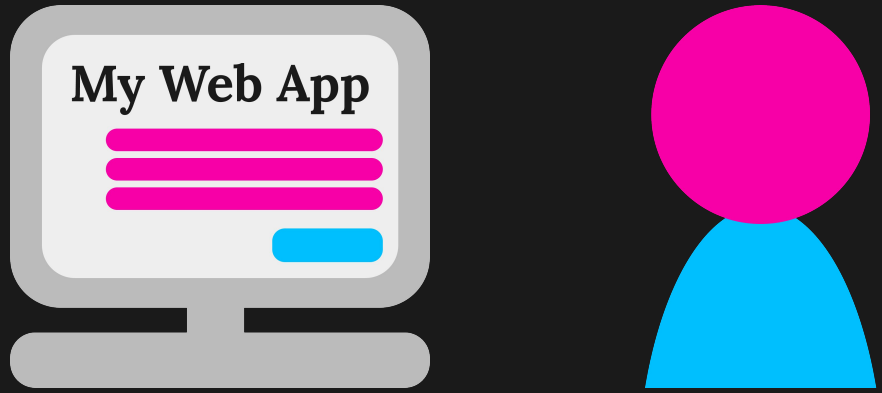
@eli@hachyderm.io

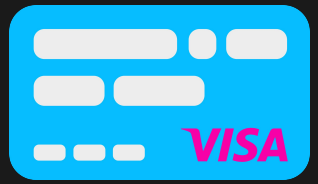

My Web App
VISA



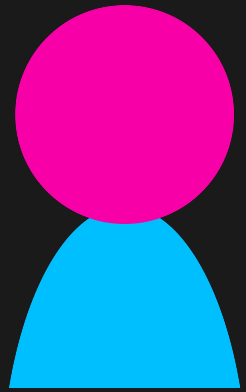


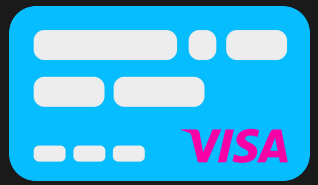
@username



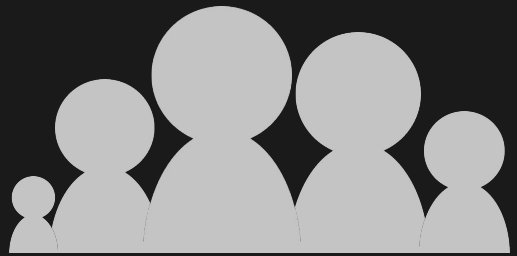


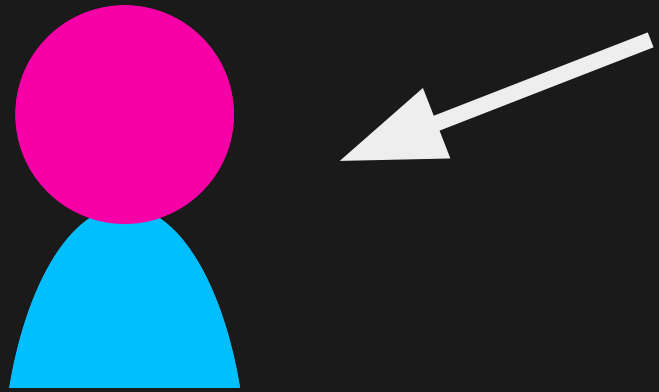
@username

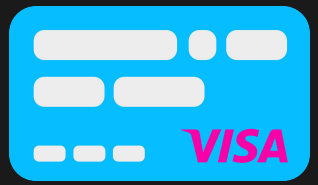




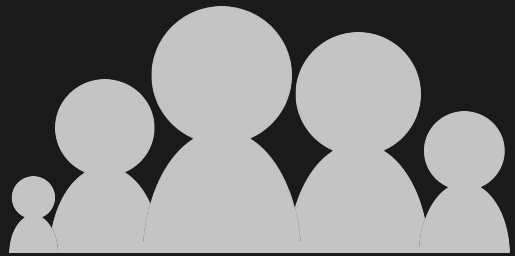
@username

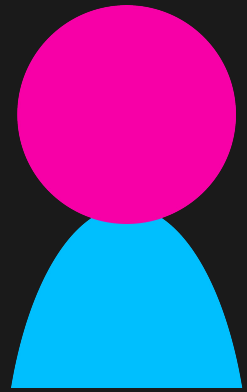


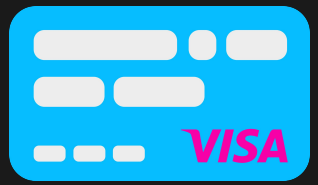




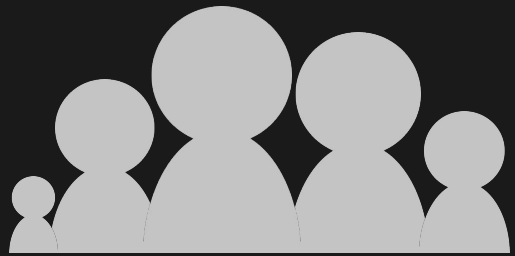
@username







@username



@eli@hachyderm.io

As a web service provider

As a web service provider

I want to verify that a given user corresponds to a given identity

As a web service provider

I want to verify that a given user corresponds to a given identity

So that they have appropriate access to data and actions based on that identity

How do we do it?

Identity is hard to define!

Identity is hard to define!

So, we use abstractions instead...

Identity is hard to define!

So, we use abstractions instead...

... but these are only ever approximate.

Username

Password

Log In

Log in

Log in with Google

Log in with Facebook

Log in with Twitter

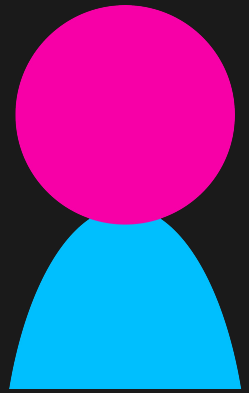
your code is
362907

@eli@hachyderm.io

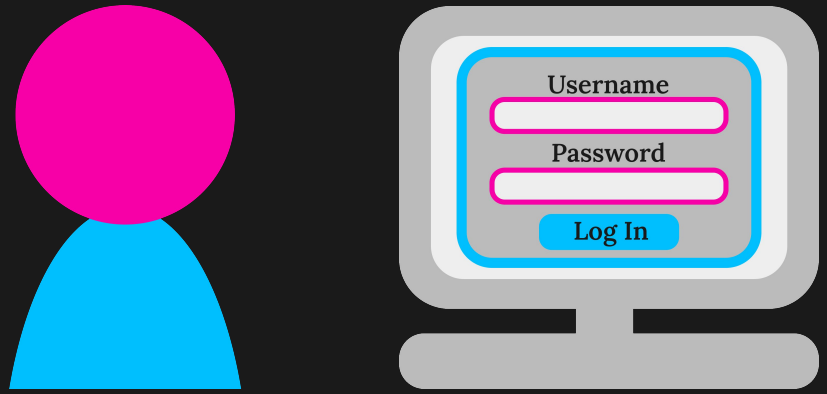
@eli@hachyderm.io

Username & password

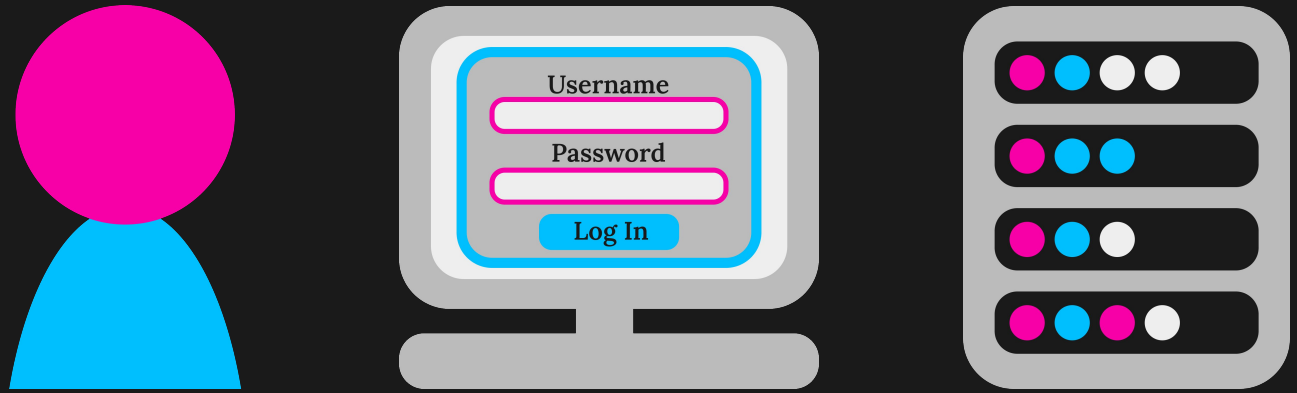
@eli@hachyderm.io

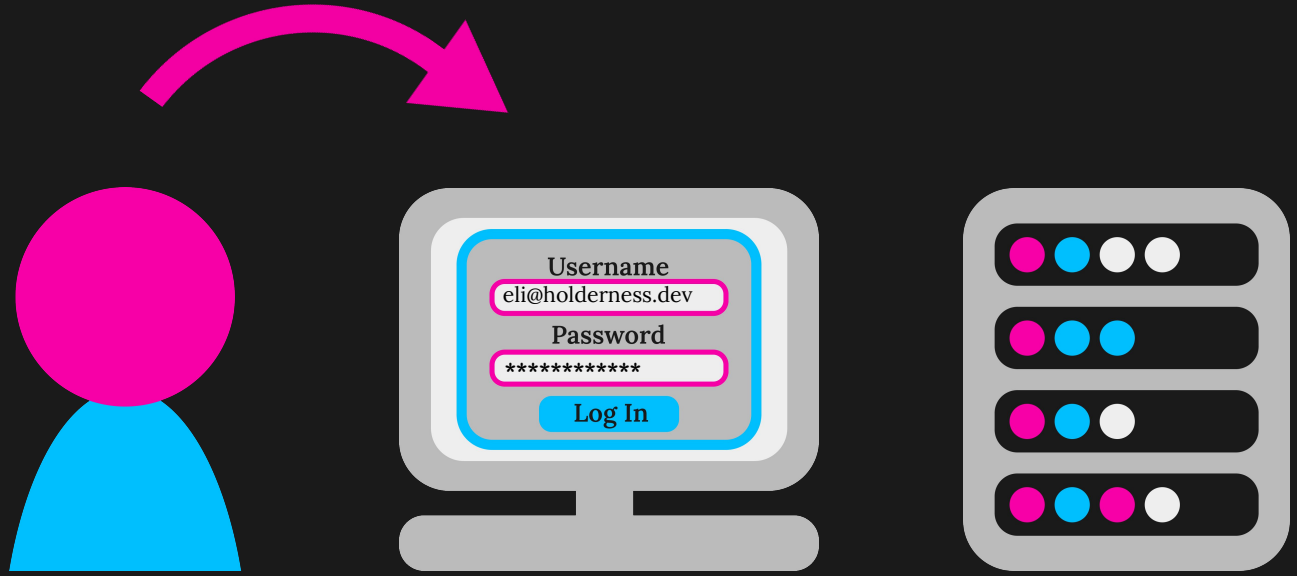


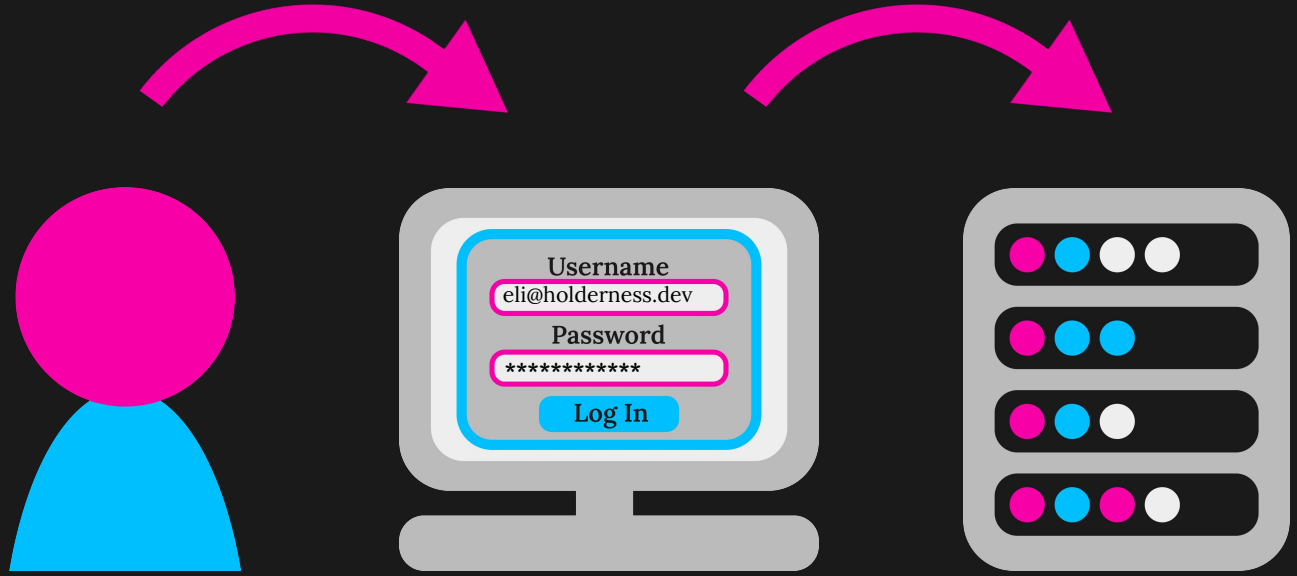
@eli@hachyderm.io

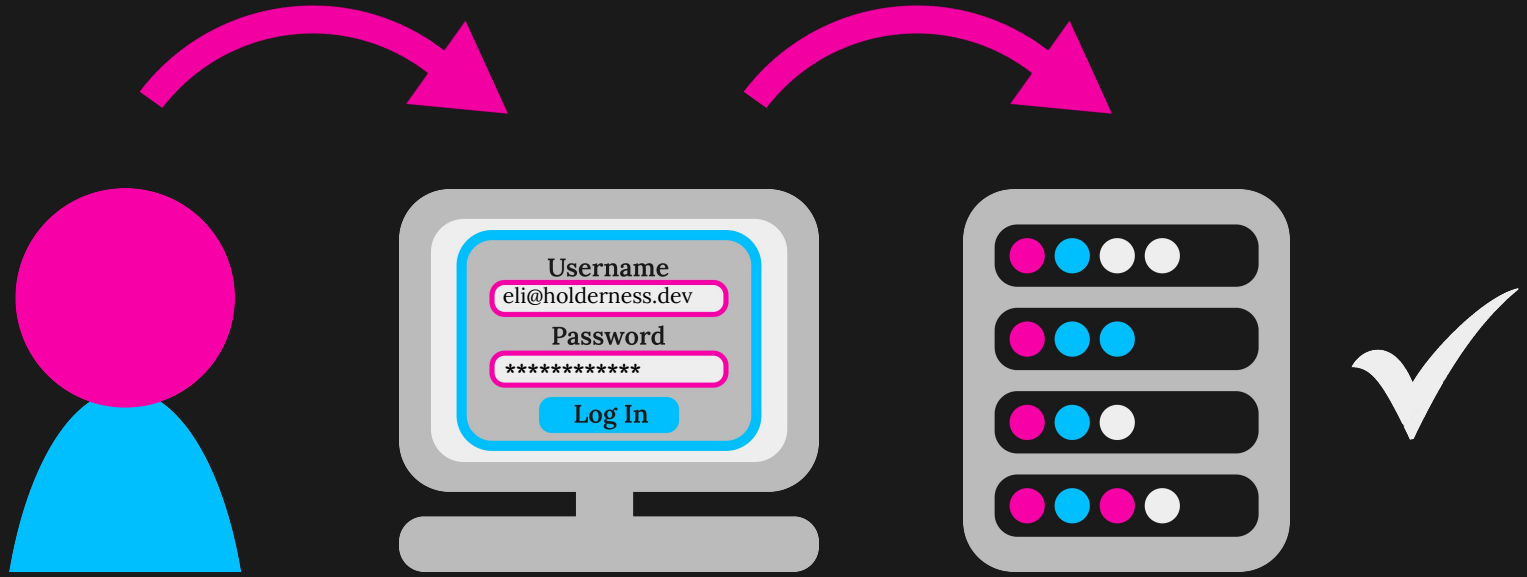


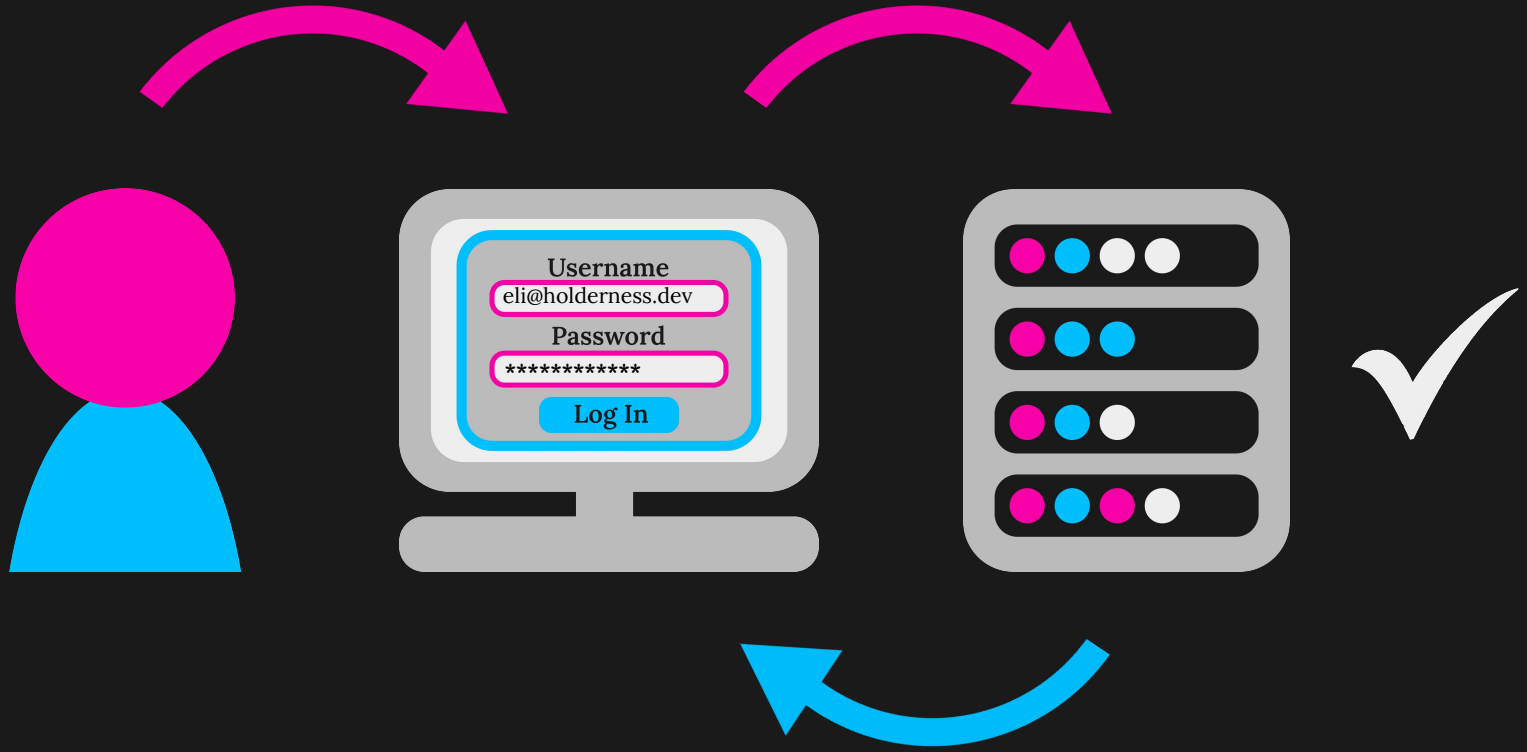
@eli@hachyderm.io

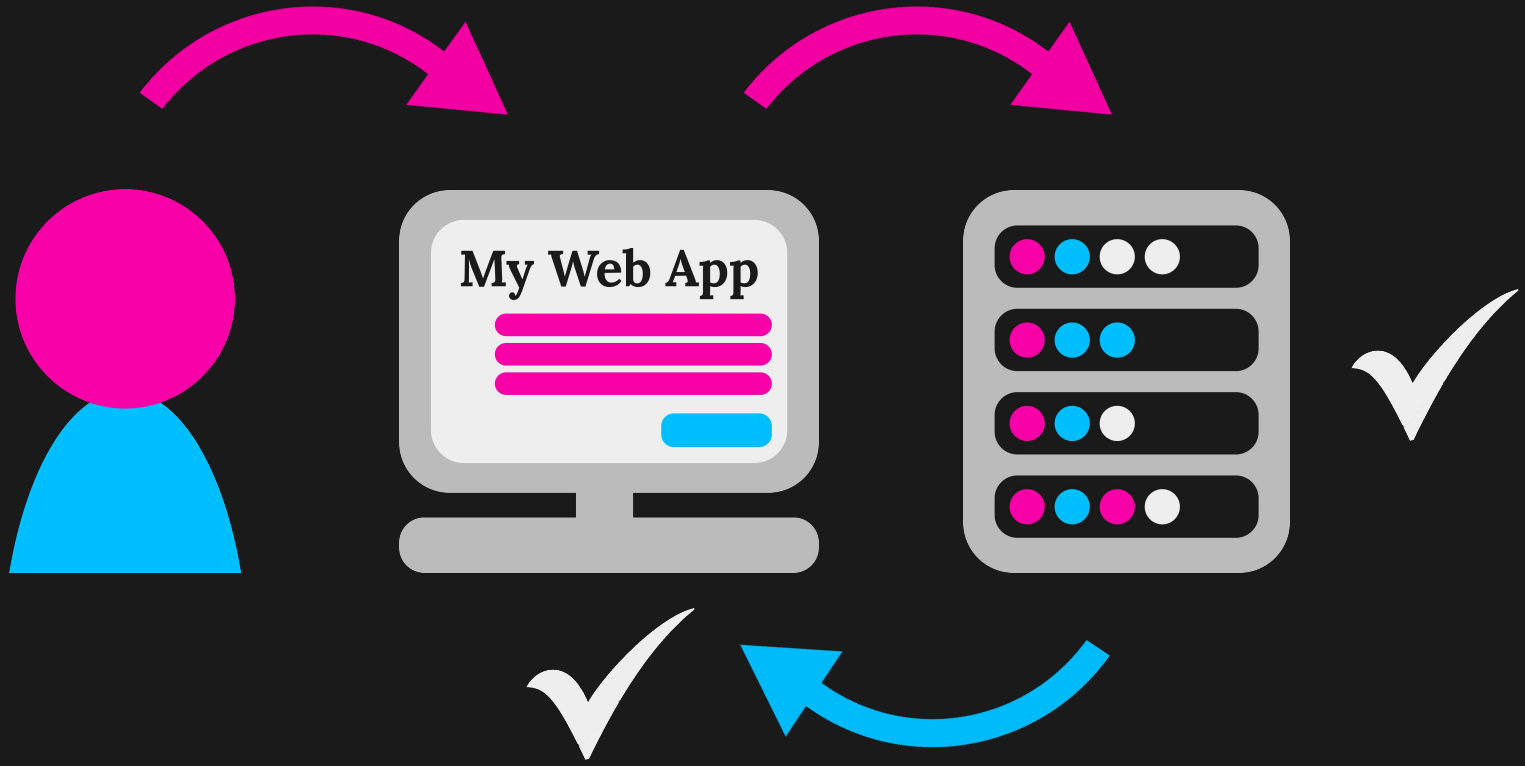

Username
Password
Log In


Username
eli@holderness.dev
Password
****
Log In


Username
eli@holderness.dev
Password
****
Log In


Username
eli@holderness.dev
Password
****
Log In


Username
eli@holderness.dev
Password
****
Log In



@eli@hachyderm.io

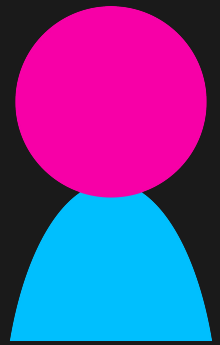
We assume that knowledge of the password is equivalent to identity.

We assume that knowledge of the password is equivalent to identity.

So, who knows it?

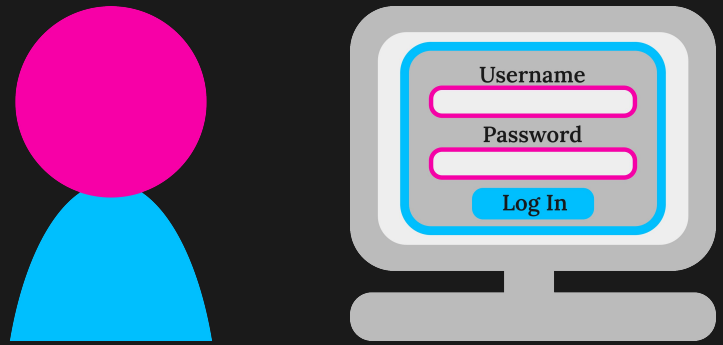
We assume that knowledge of the password is equivalent to identity.

So, who knows it?



We assume that knowledge of the password is equivalent to identity.

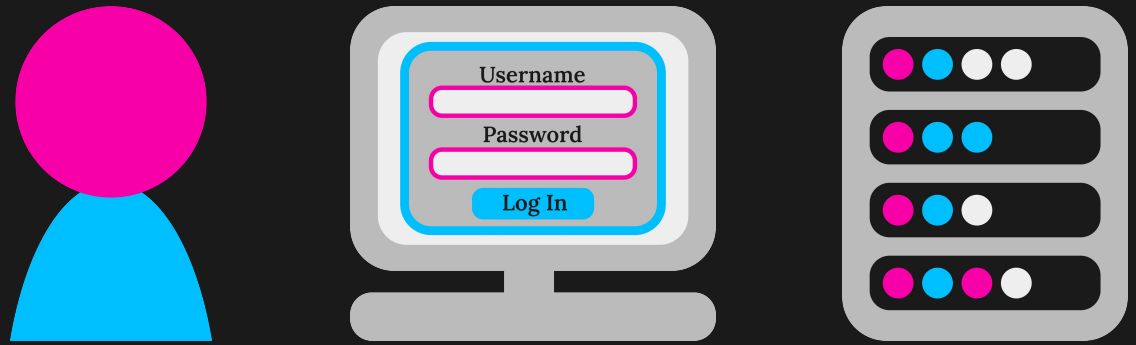
So, who knows it?



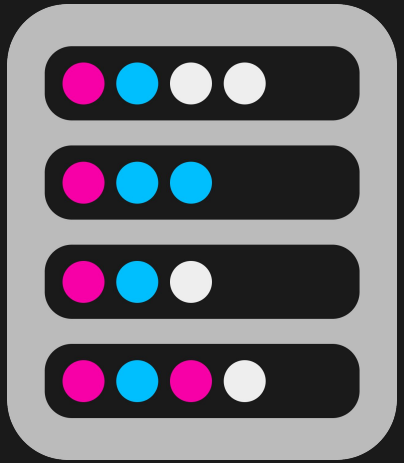
@eli@hachyderm.io

We assume that knowledge of the password is equivalent to identity.

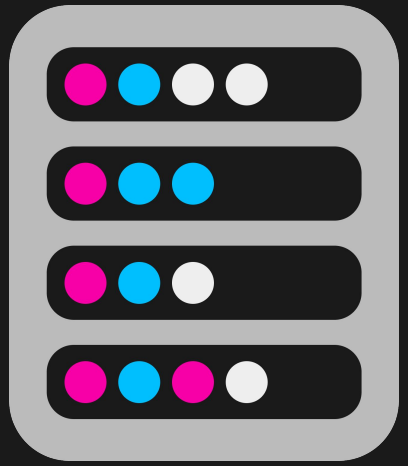
So, who knows it?



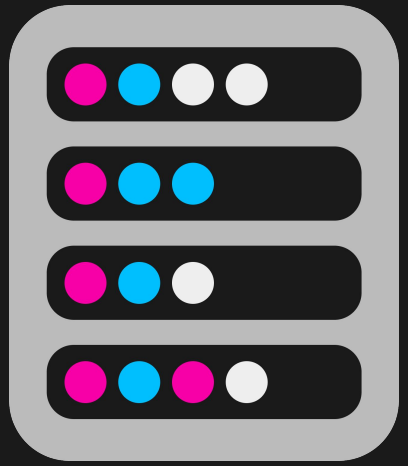
@eli@hachyderm.io



@eli@hachyderm.io

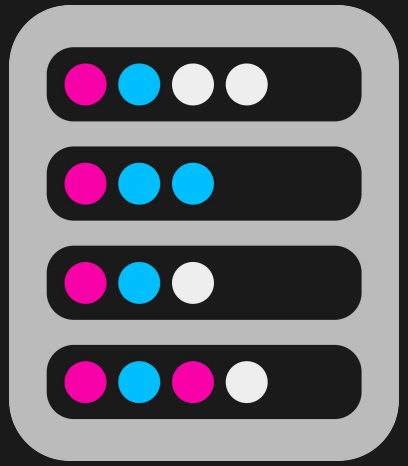


... must be able to verify the password



... must be able to verify the password

... must not allow the password to be discovered



username: eli
password: Eli’sPassword

@eli@hachyderm.io

username
password: assword

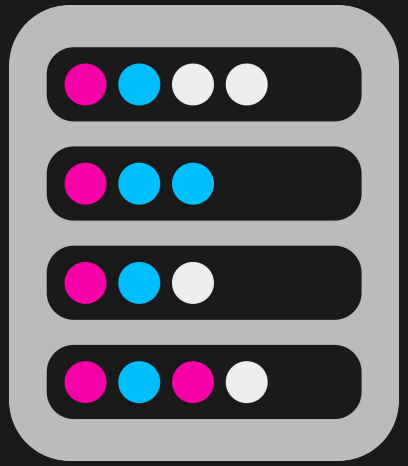
@eli@hachyderm.io

hash(“Eli’sPassword”)

=

“8sit56&toh3a”

@eli@hachyderm.io



username: eli
hashed_password: 8sit56&toh3a

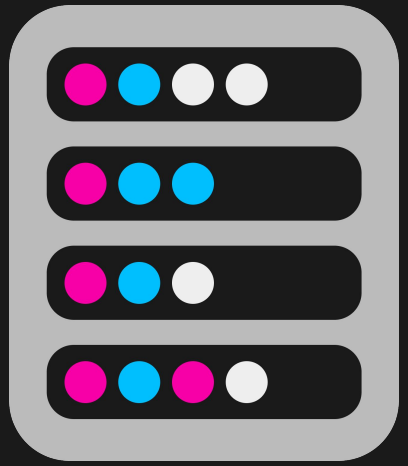
@eli@hachyderm.io

hash(“Eli’sPassword7af”)

=

“9&GDSo8fgLIHD”

@eli@hachyderm.io



username: eli
salted_hash: 7af$9&GDSo8fgLIHD

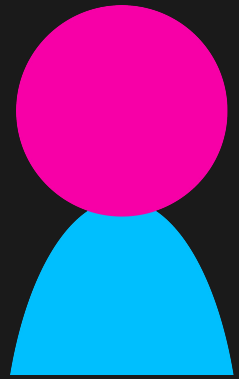
@eli@hachyderm.io

… but, for most passwords,
if someone gets hold
of your salted hash it's
still game over

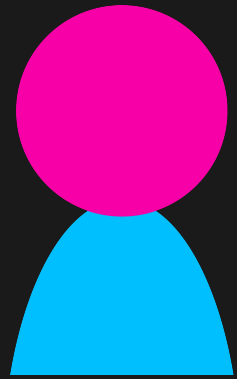
@eli@hachyderm.io

9&GDSo8fgLIHD

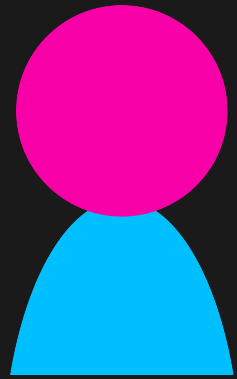
@eli@hachyderm.io



@eli@hachyderm.io

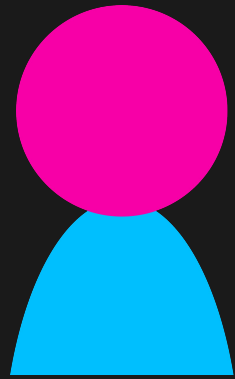


... can forget their password



... can forget their password

... can be tricked into giving away their password



... can forget their password

... can be tricked into giving away their password

... can be tempted to re-use a strong password

@eli@hachyderm.io

So, how can we do better?

Multi-factor Authentication

@eli@hachyderm.io

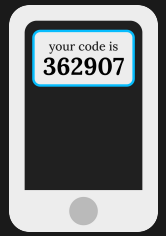
@eli@hachyderm.io

To mitigate the risk of one abstraction failing, we require multiple successes

things you know

@eli@hachyderm.io

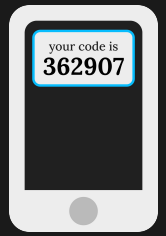
things you know



things you have

@eli@hachyderm.io

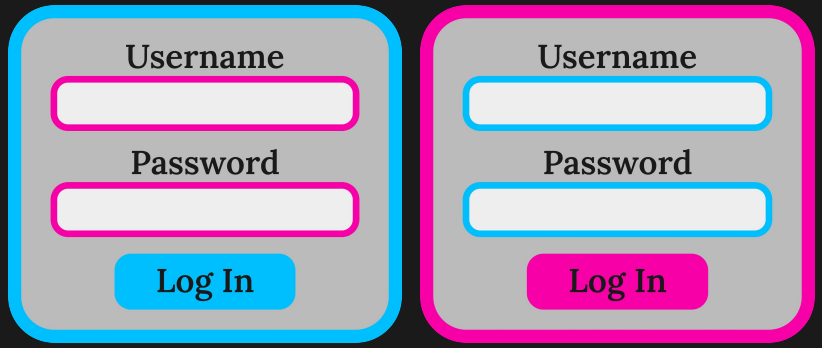
things you know



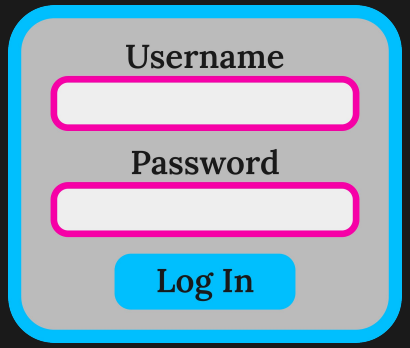
things you have

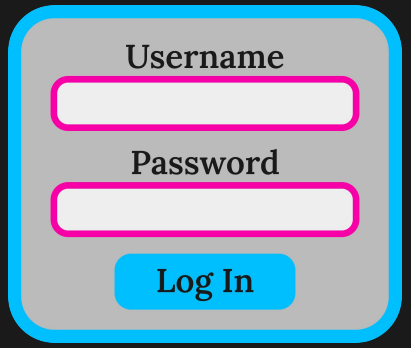
things you are

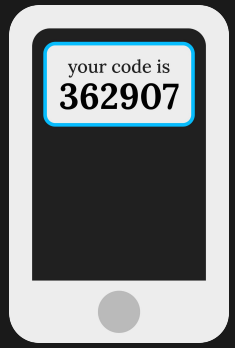
@eli@hachyderm.io



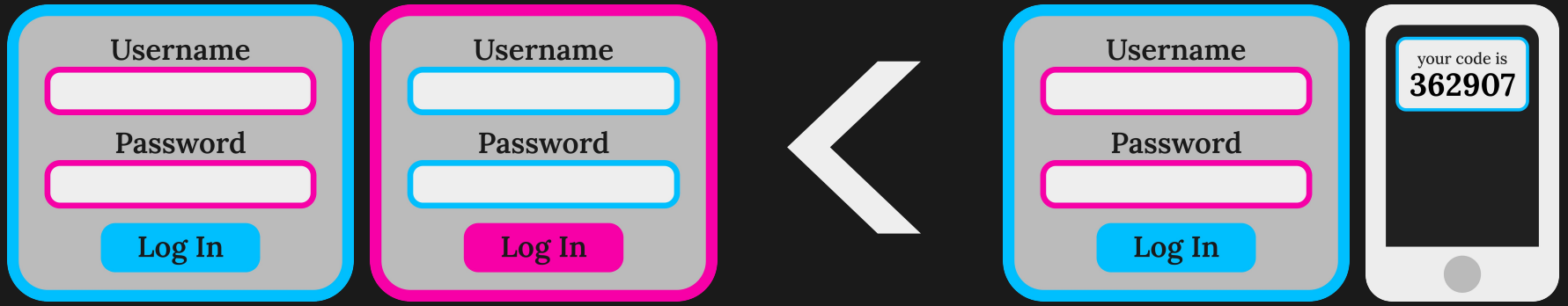
@eli@hachyderm.io







@eli@hachyderm.io

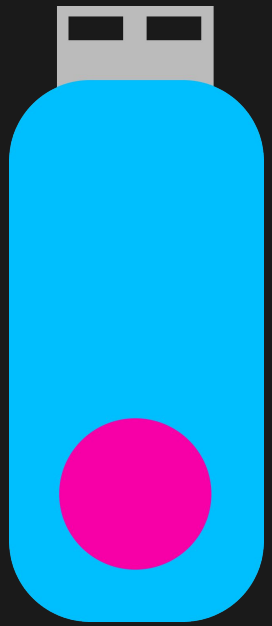


@eli@hachyderm.io

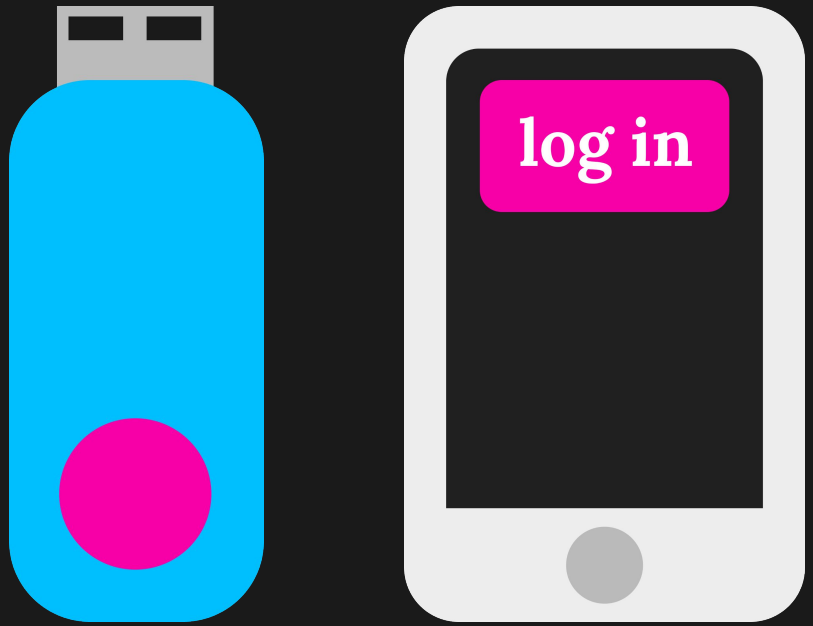
So, what are some options for MFA?

Hardware authentication tokens

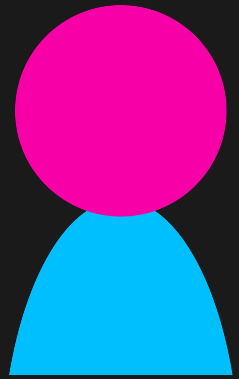
@eli@hachyderm.io



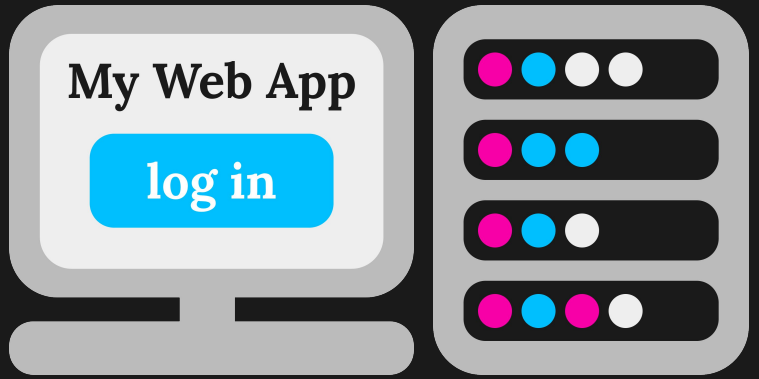
@eli@hachyderm.io


log in

public key cryptography



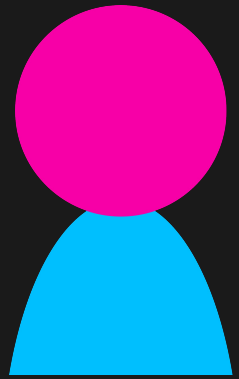
private key



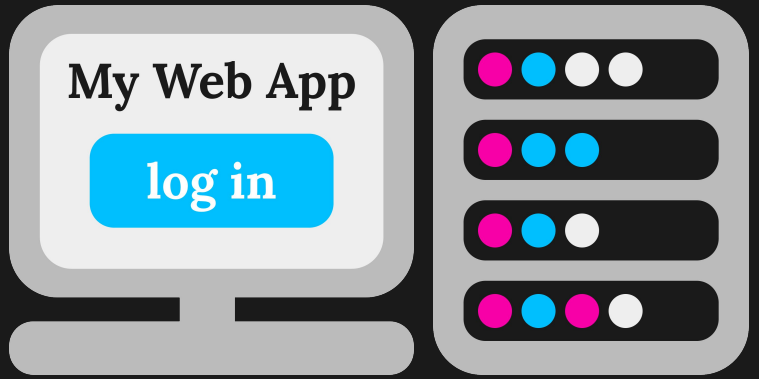
public key

@eli@hachyderm.io

public key cryptography



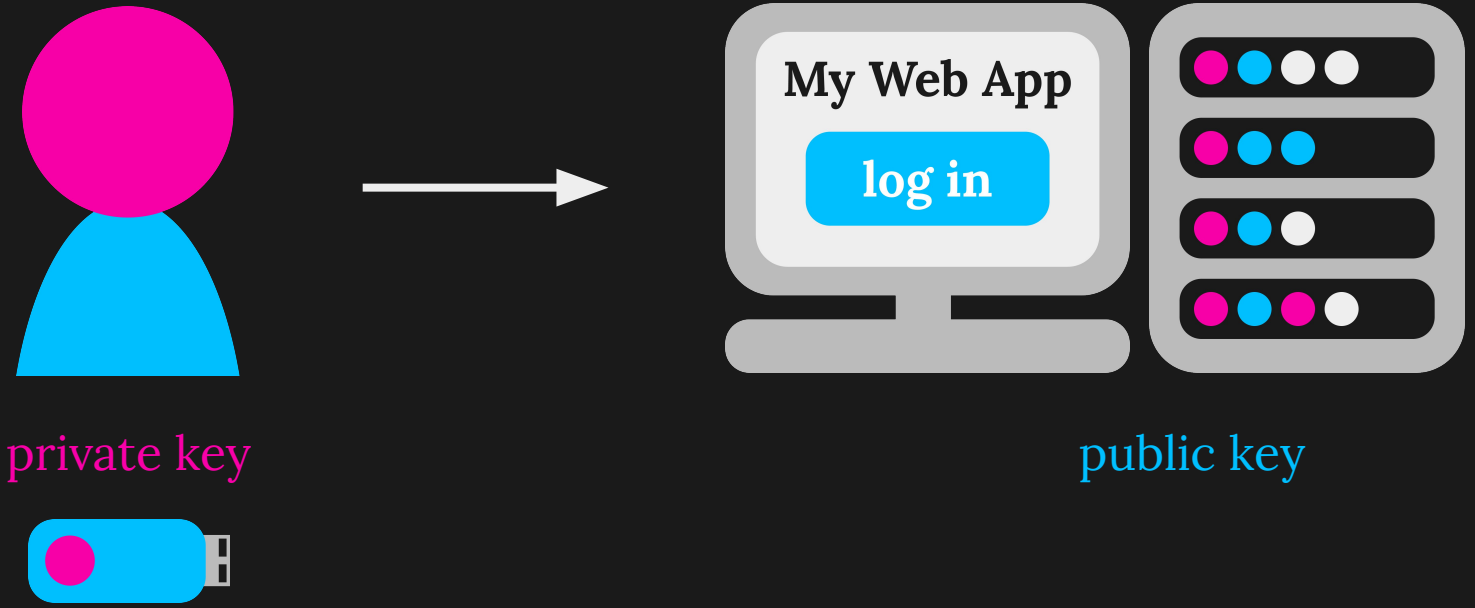
private key



public key

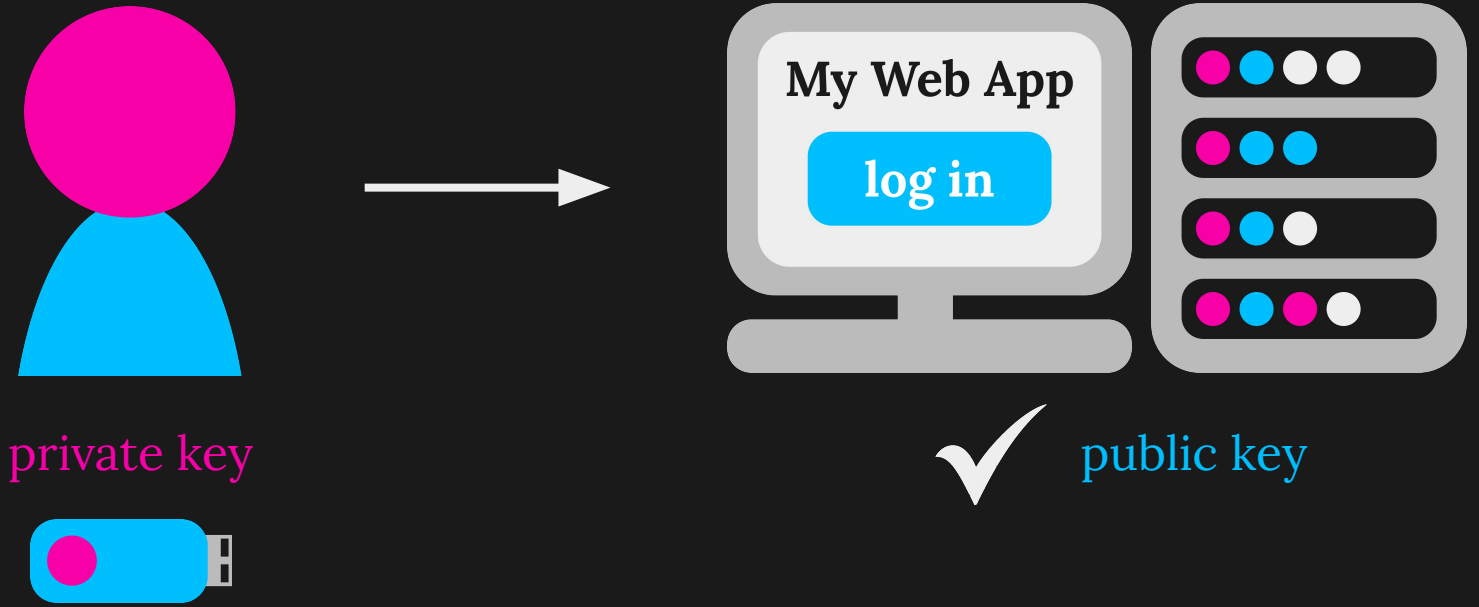
@eli@hachyderm.io

public key cryptography



@eli@hachyderm.io

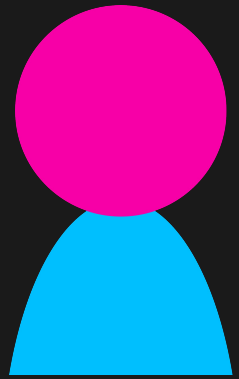
public key cryptography

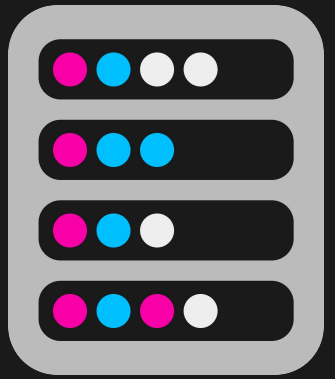


@eli@hachyderm.io

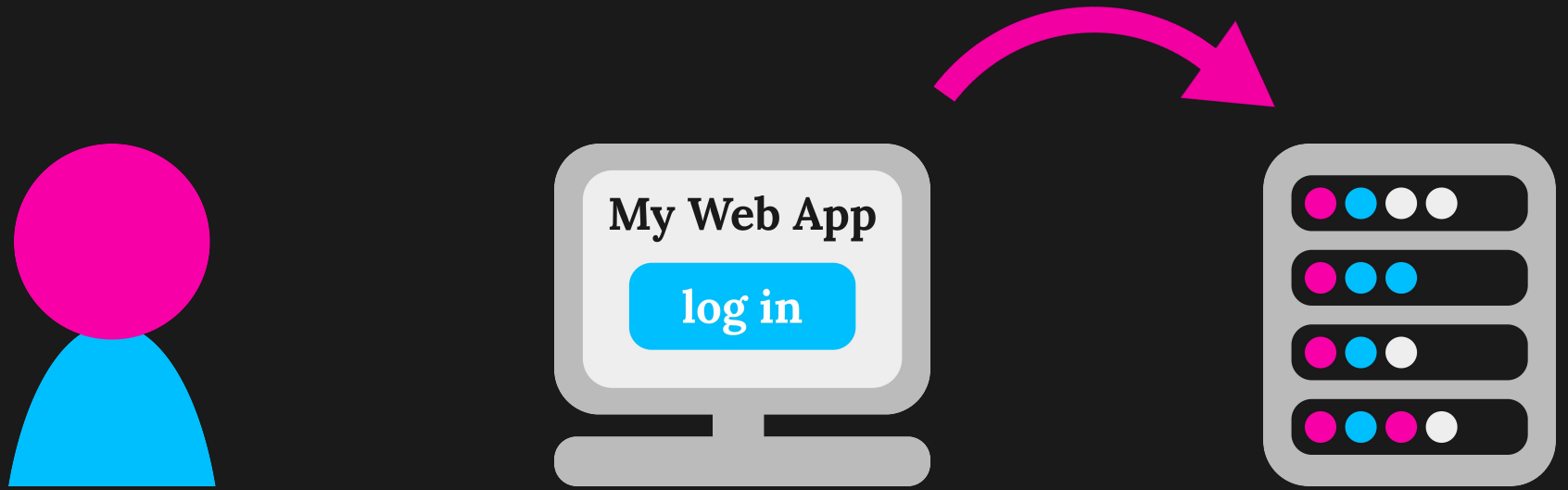
public key cryptography lets you prove ownership of a secret without giving away that secret

@eli@hachyderm.io

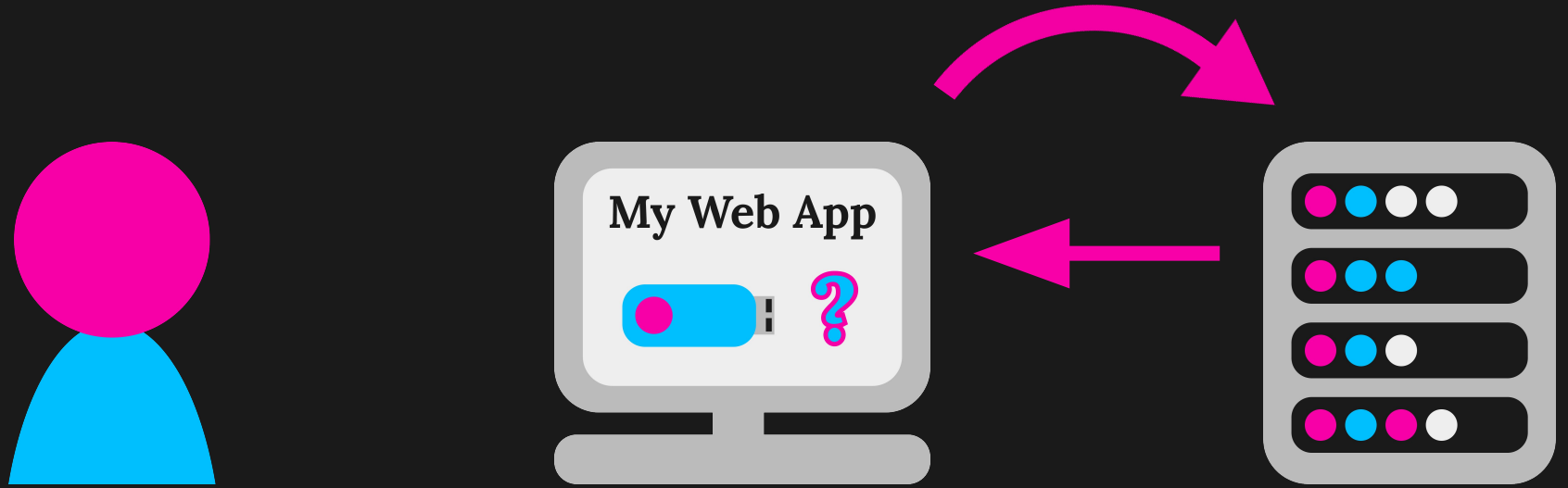


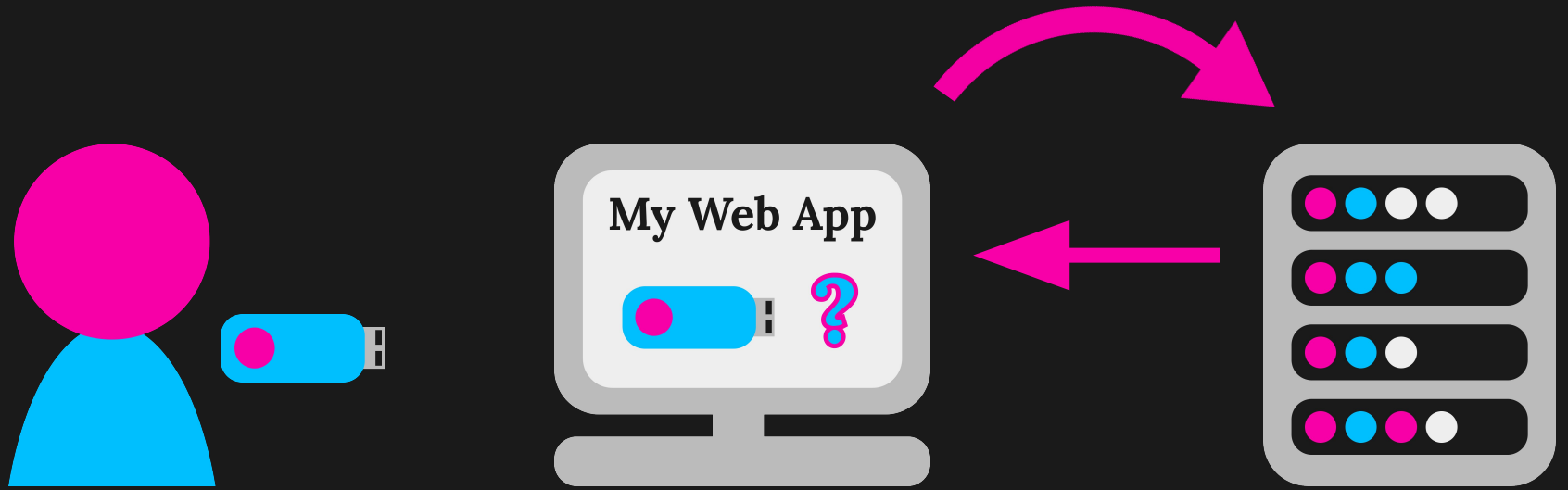


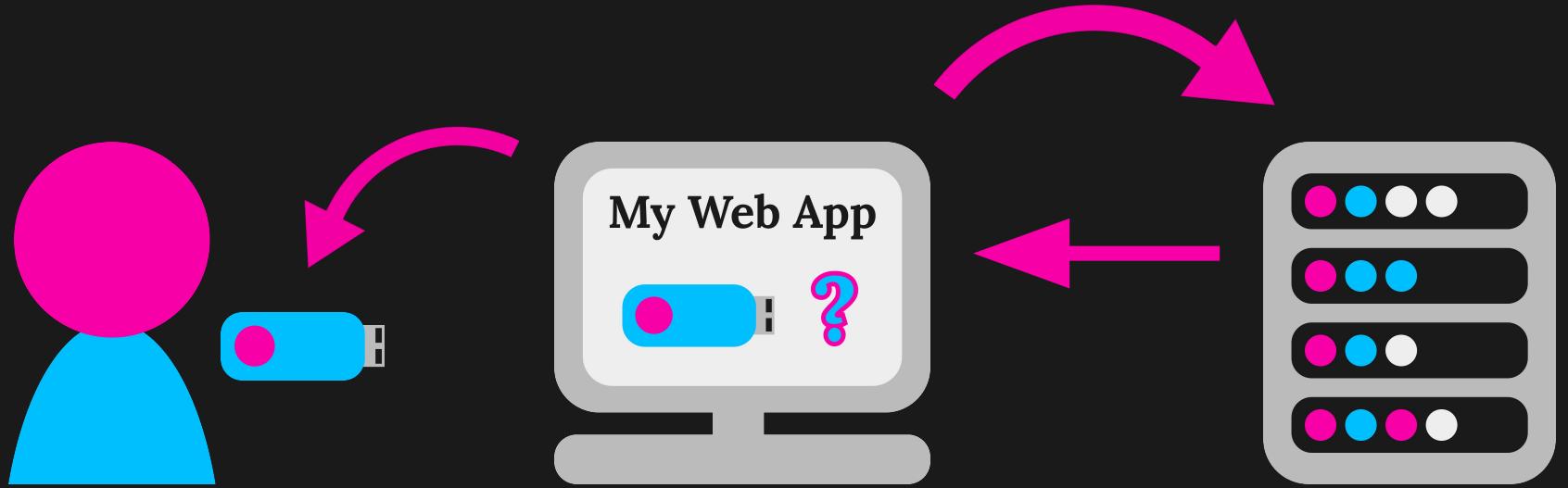
@eli@hachyderm.io


My Web App
log in

@eli@hachyderm.io


My Web App


My Web App
?



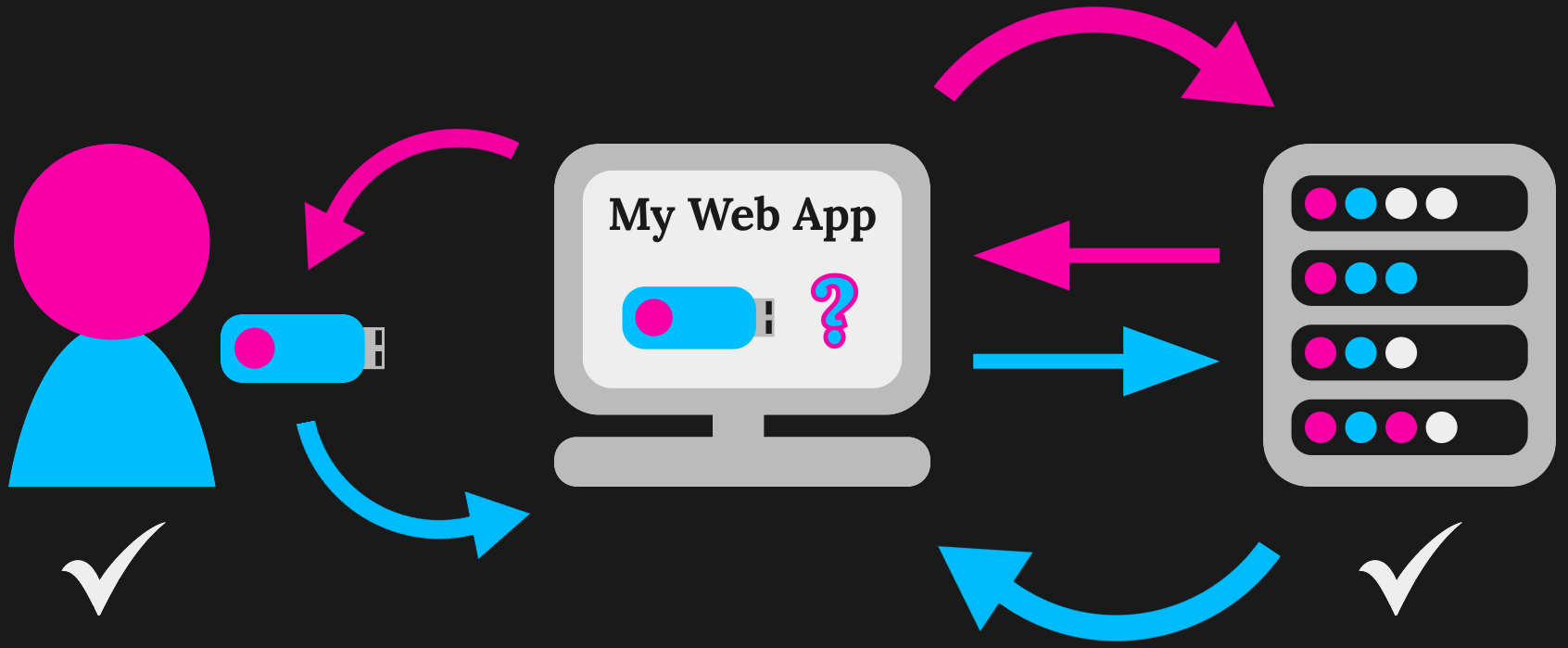
@eli@hachyderm.io


My Web App
?

@eli@hachyderm.io

@eli@hachyderm.io

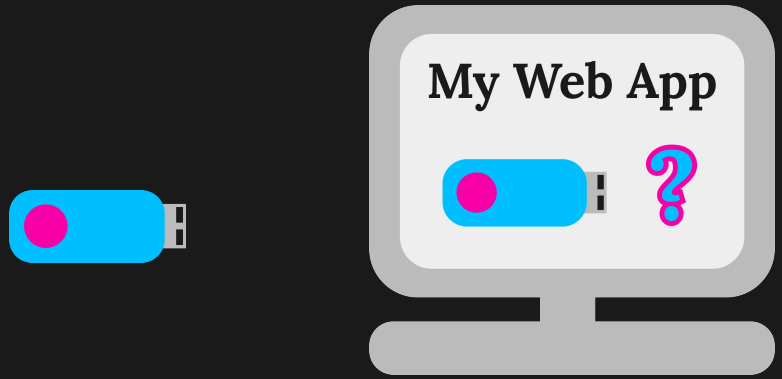
@eli@hachyderm.io



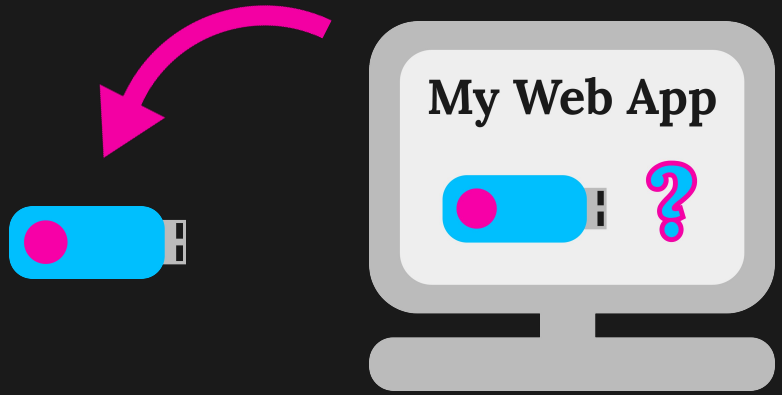
@eli@hachyderm.io

@eli@hachyderm.io

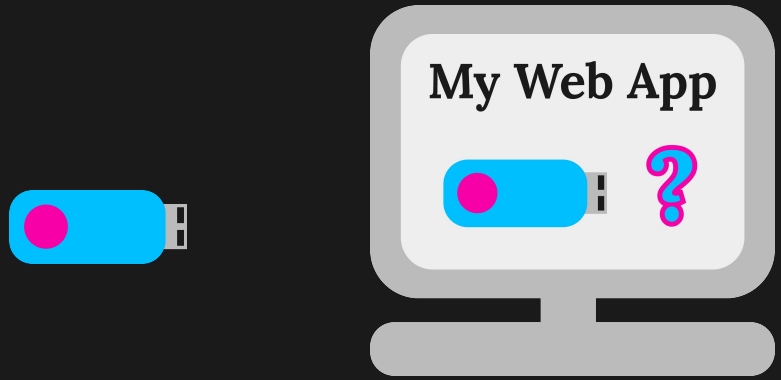
the device knows about the web service



the device knows about the web service

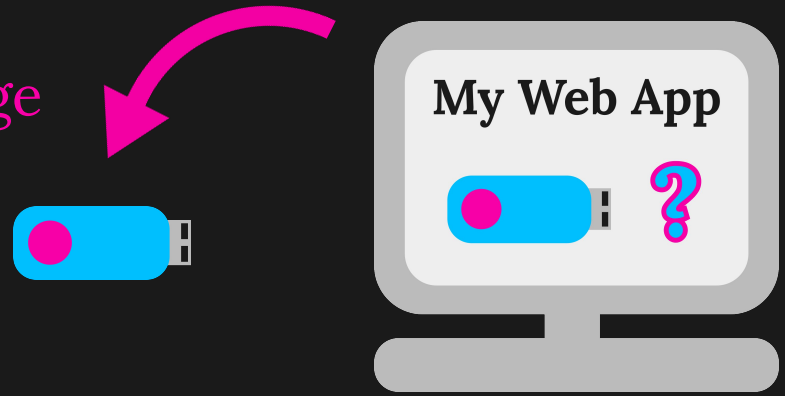


the private key never leaves the device

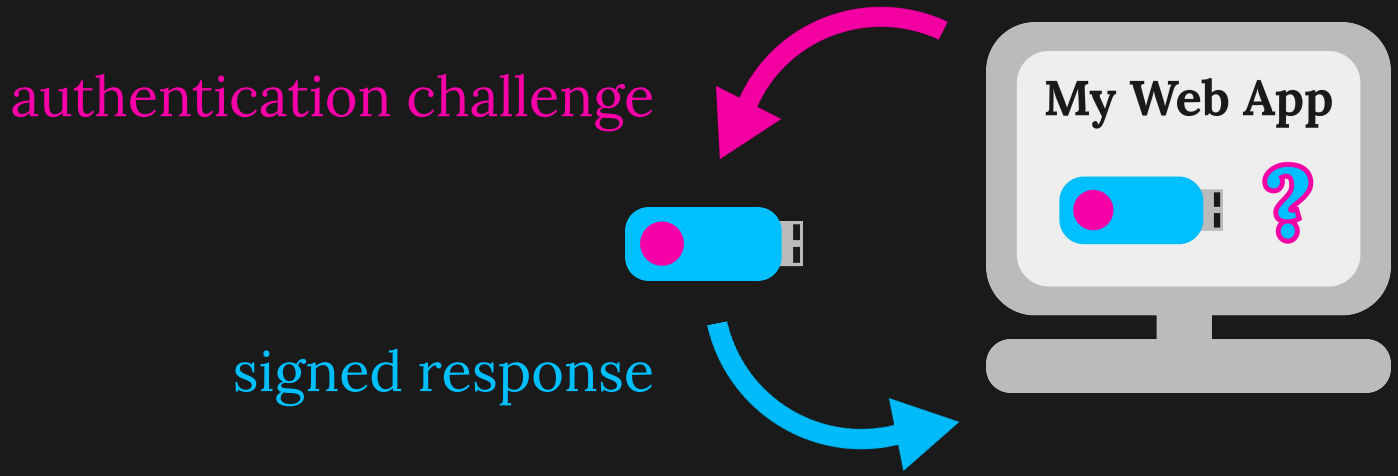


the private key never leaves the device

authentication challenge



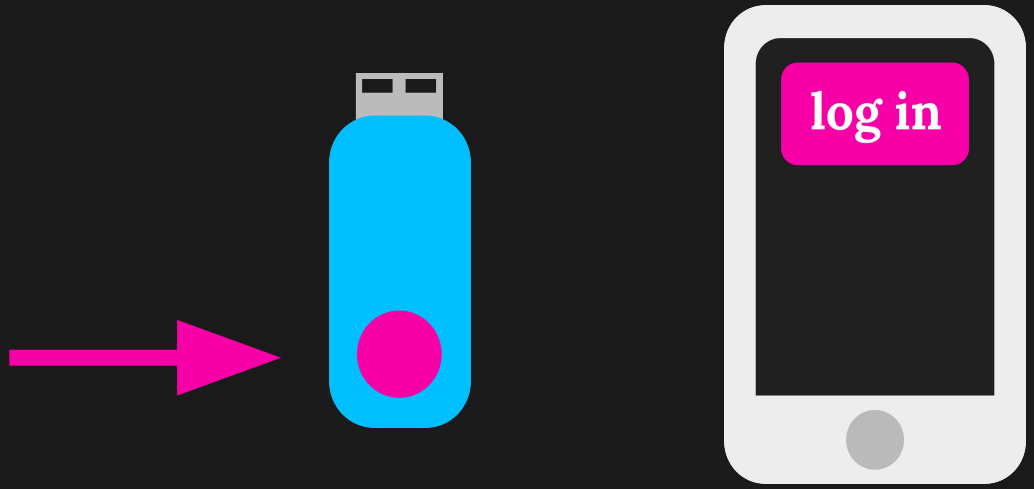
the private key never leaves the device



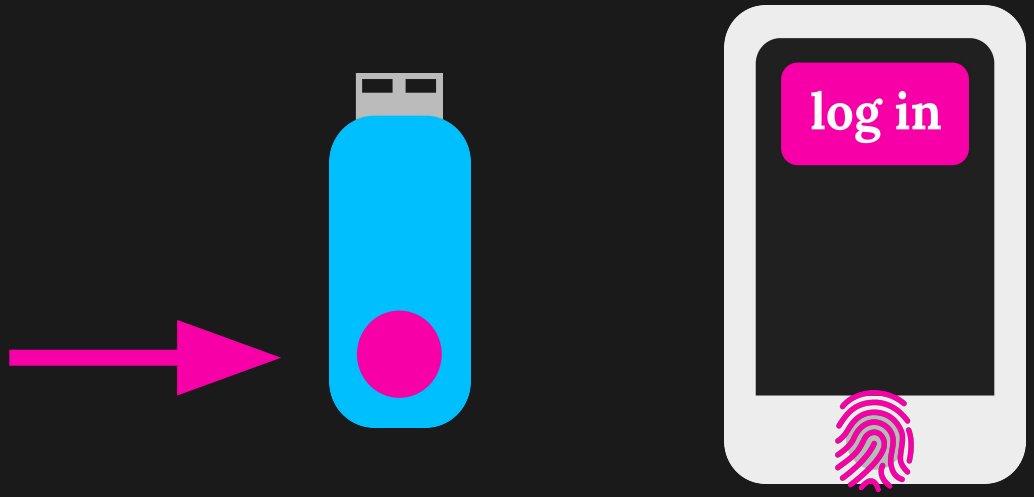
@eli@hachyderm.io

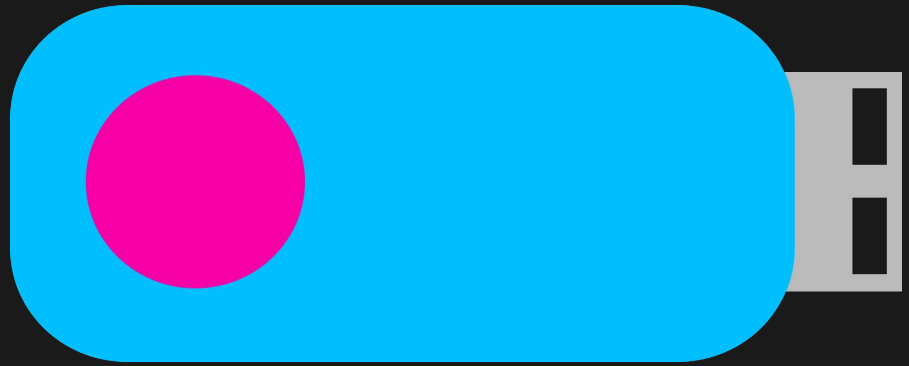
the device requires user presence

the device requires user presence



the device requires user presence





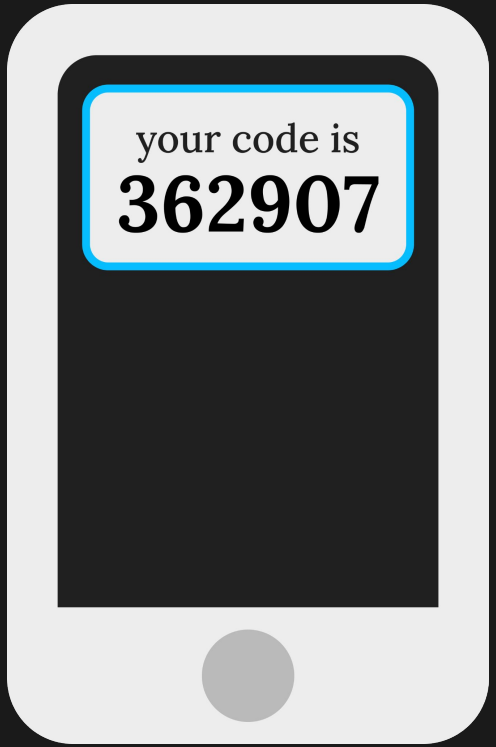
@eli@hachyderm.io

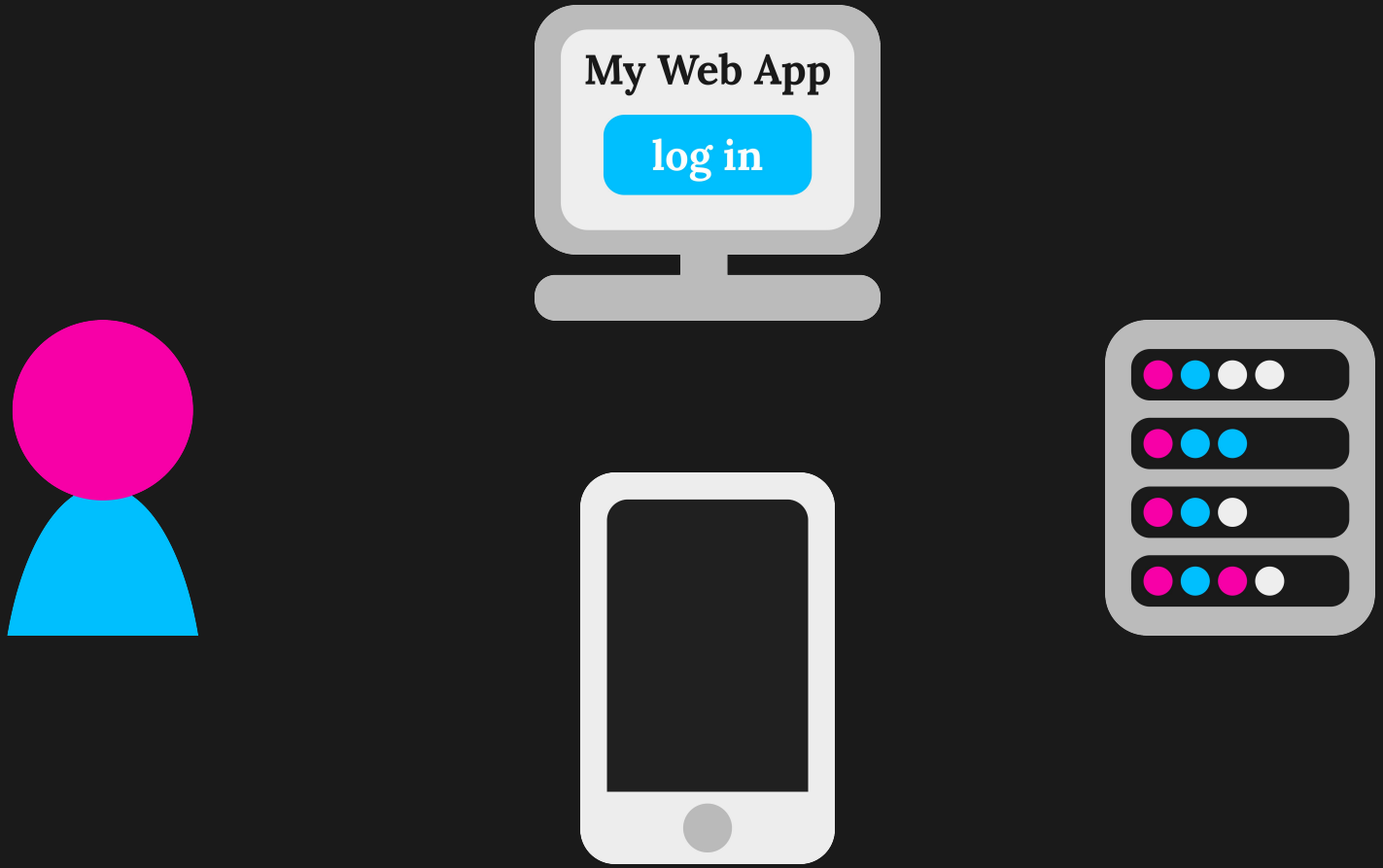
@eli@hachyderm.io

@eli@hachyderm.io

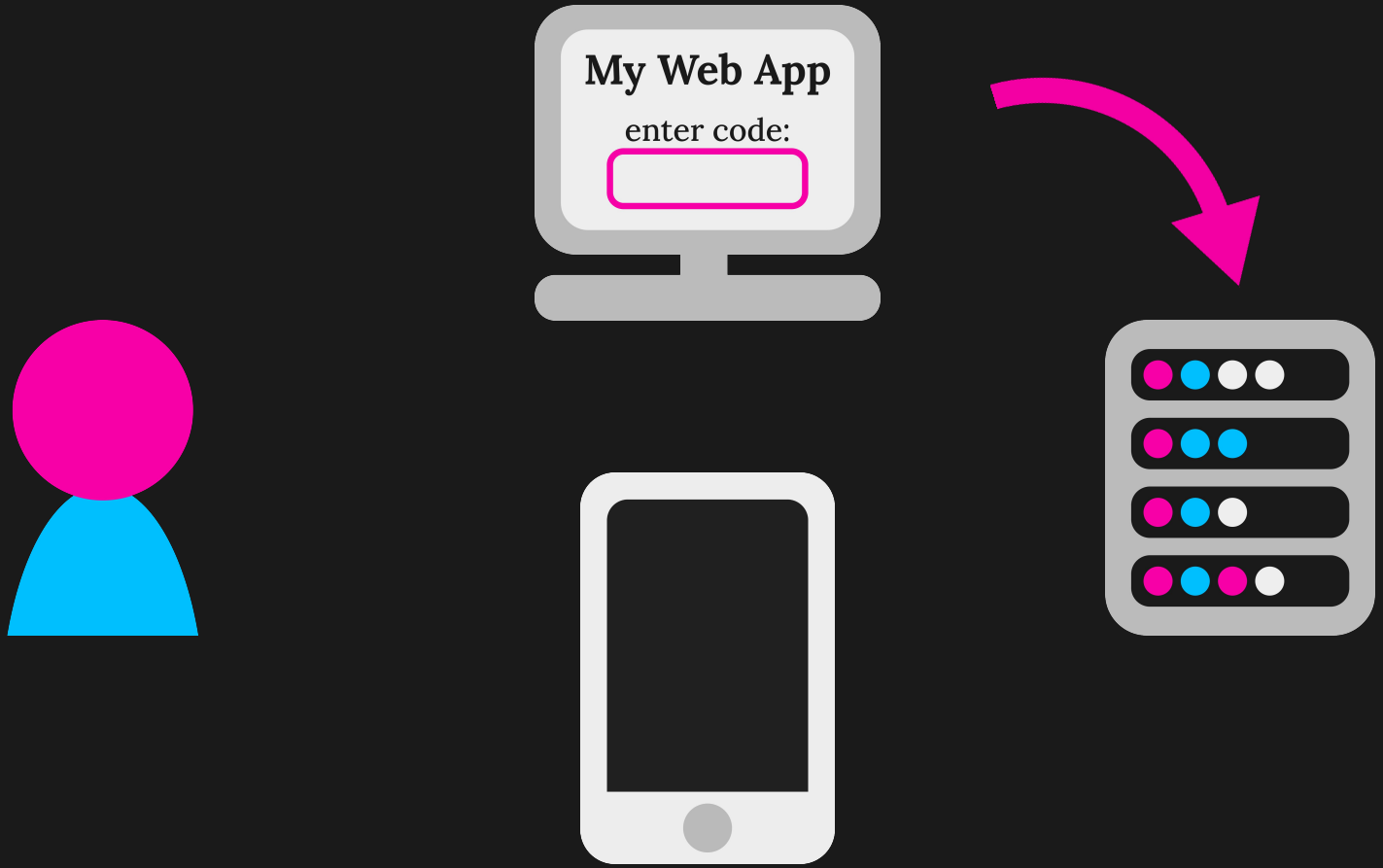
One-time passwords

@eli@hachyderm.io

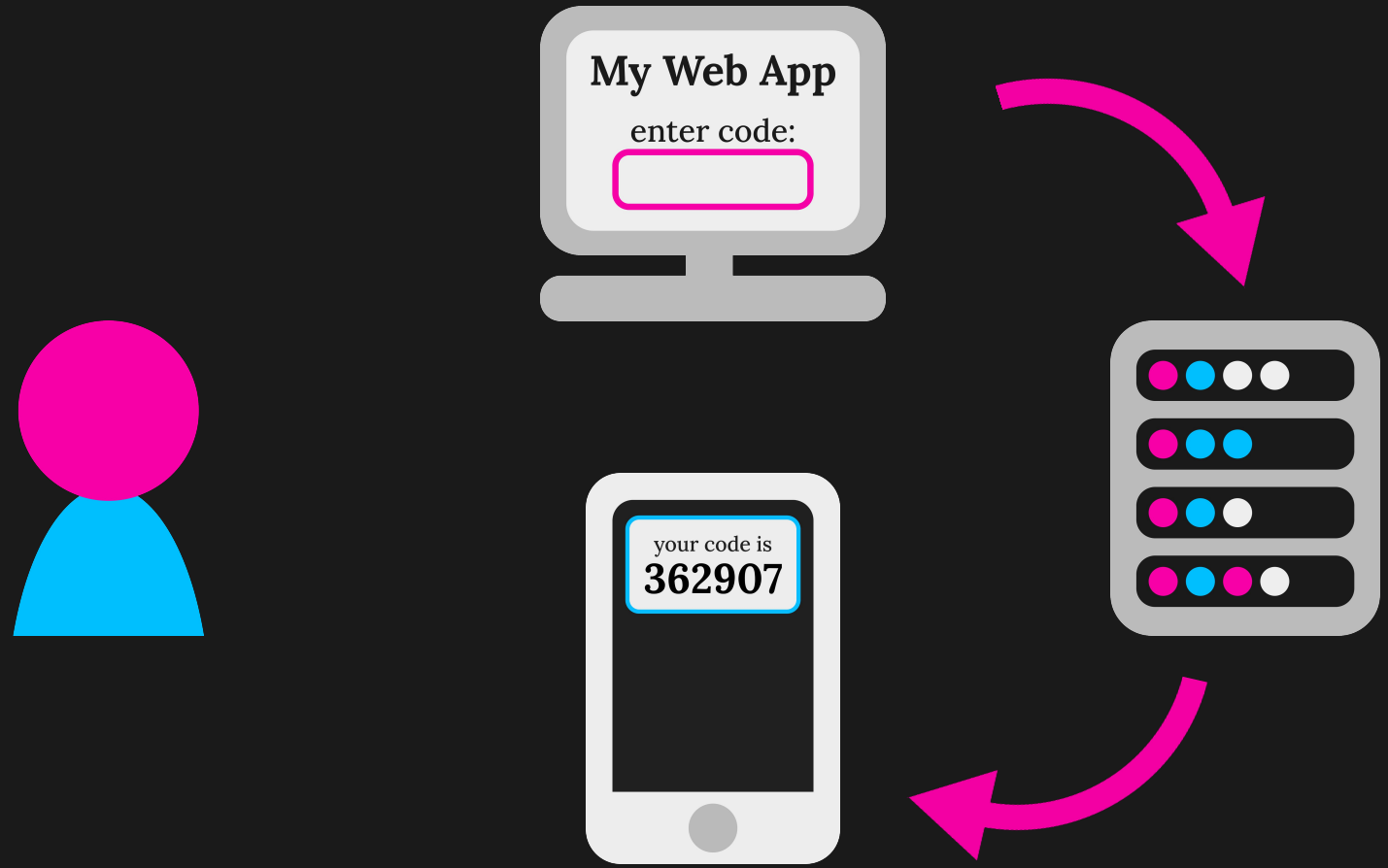

your code is
362907



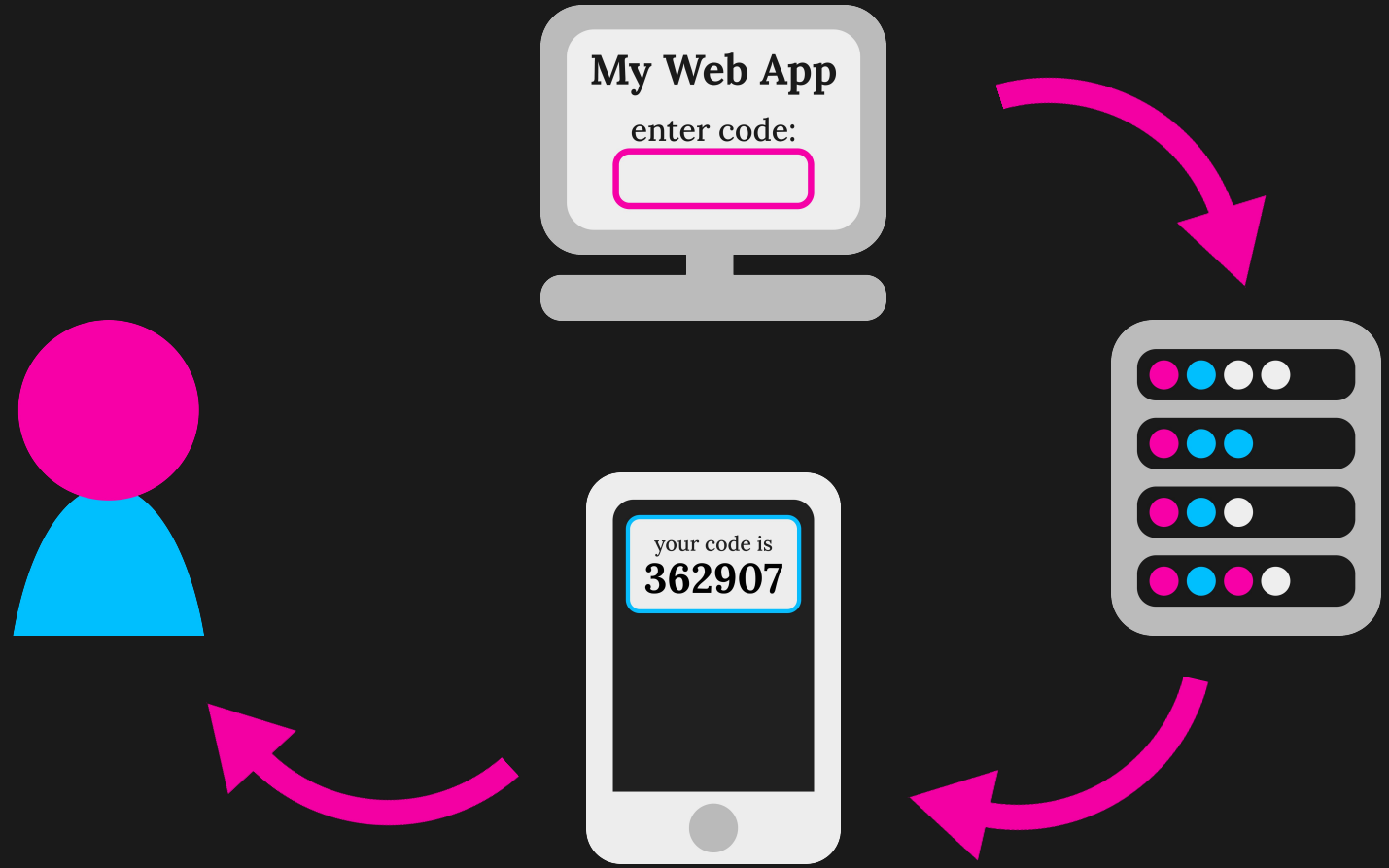
@eli@hachyderm.io



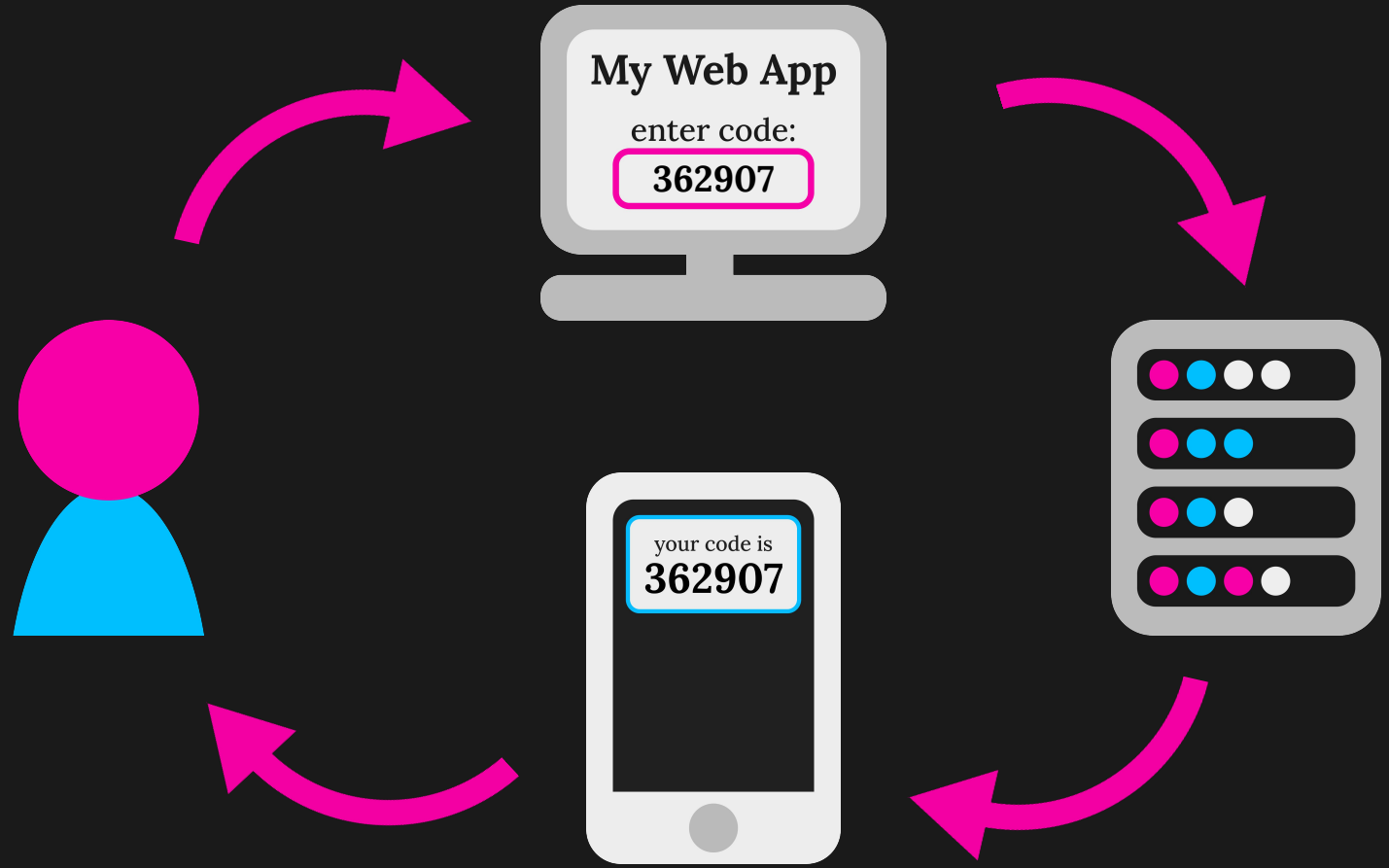
@eli@hachyderm.io



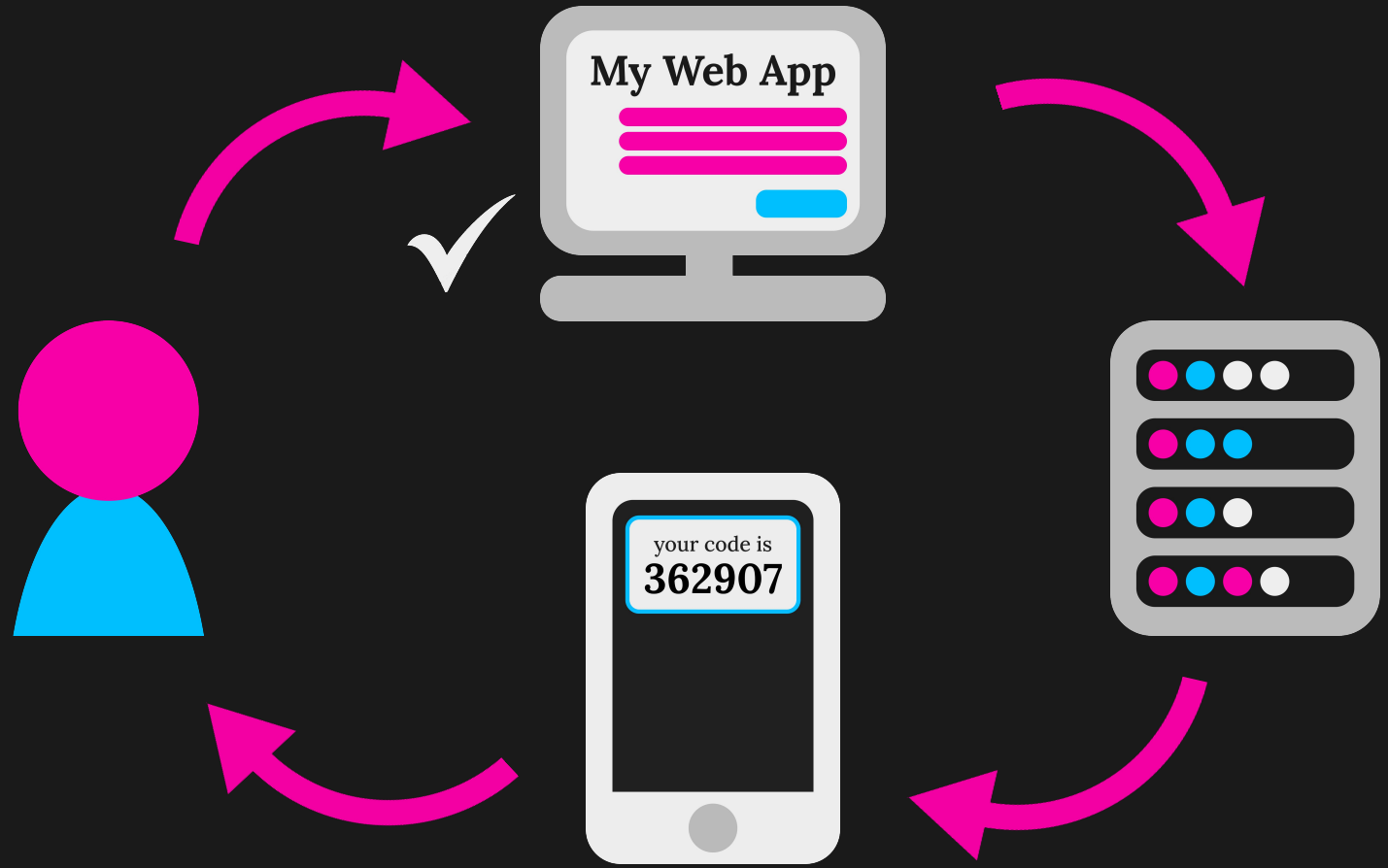
@eli@hachyderm.io



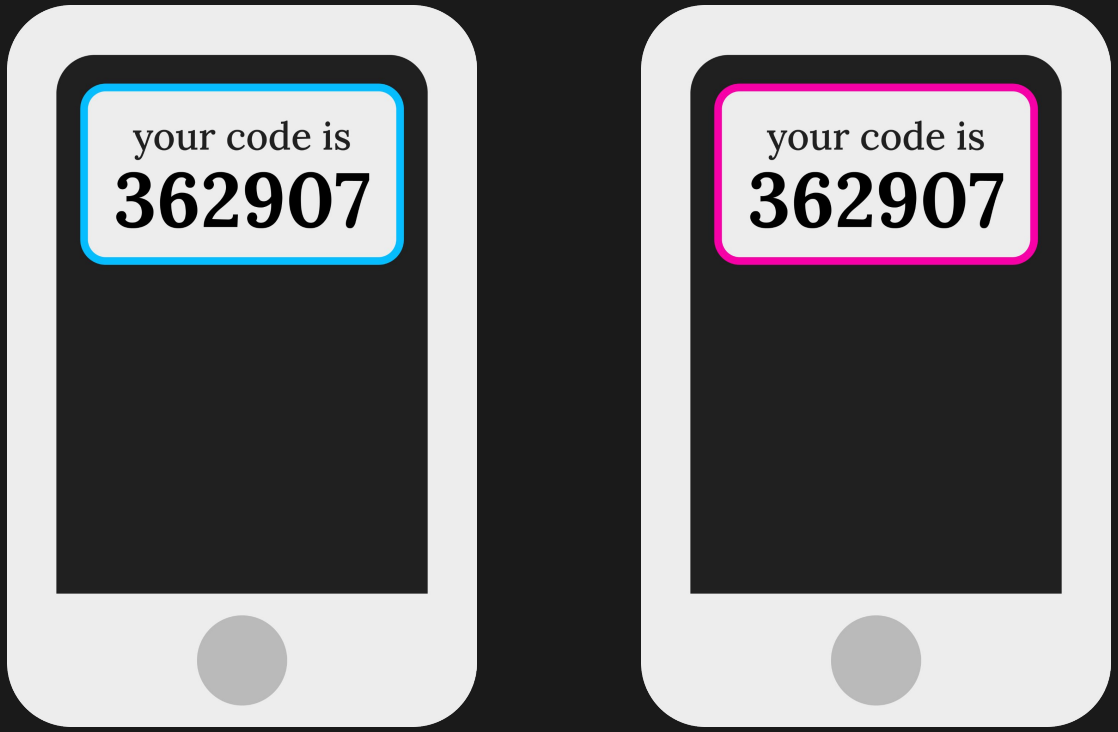
@eli@hachyderm.io

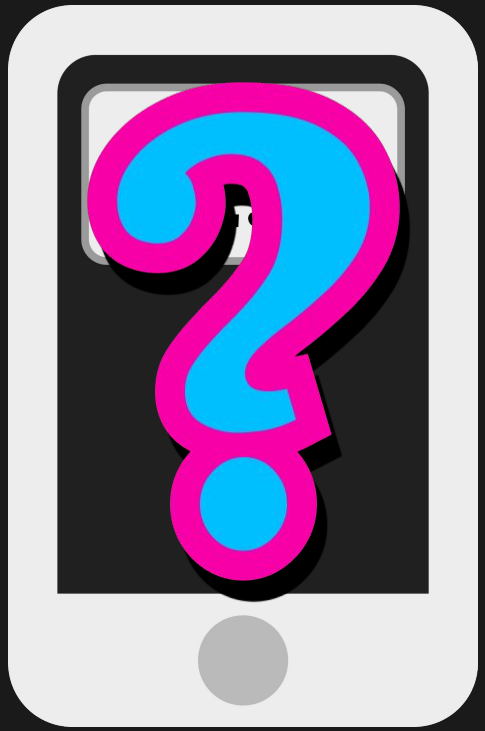


@eli@hachyderm.io



@eli@hachyderm.io


your code is
362907
your code is
362907

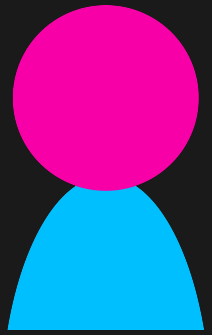


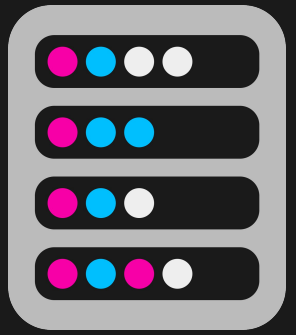
@eli@hachyderm.io

‘Magic Link’ login

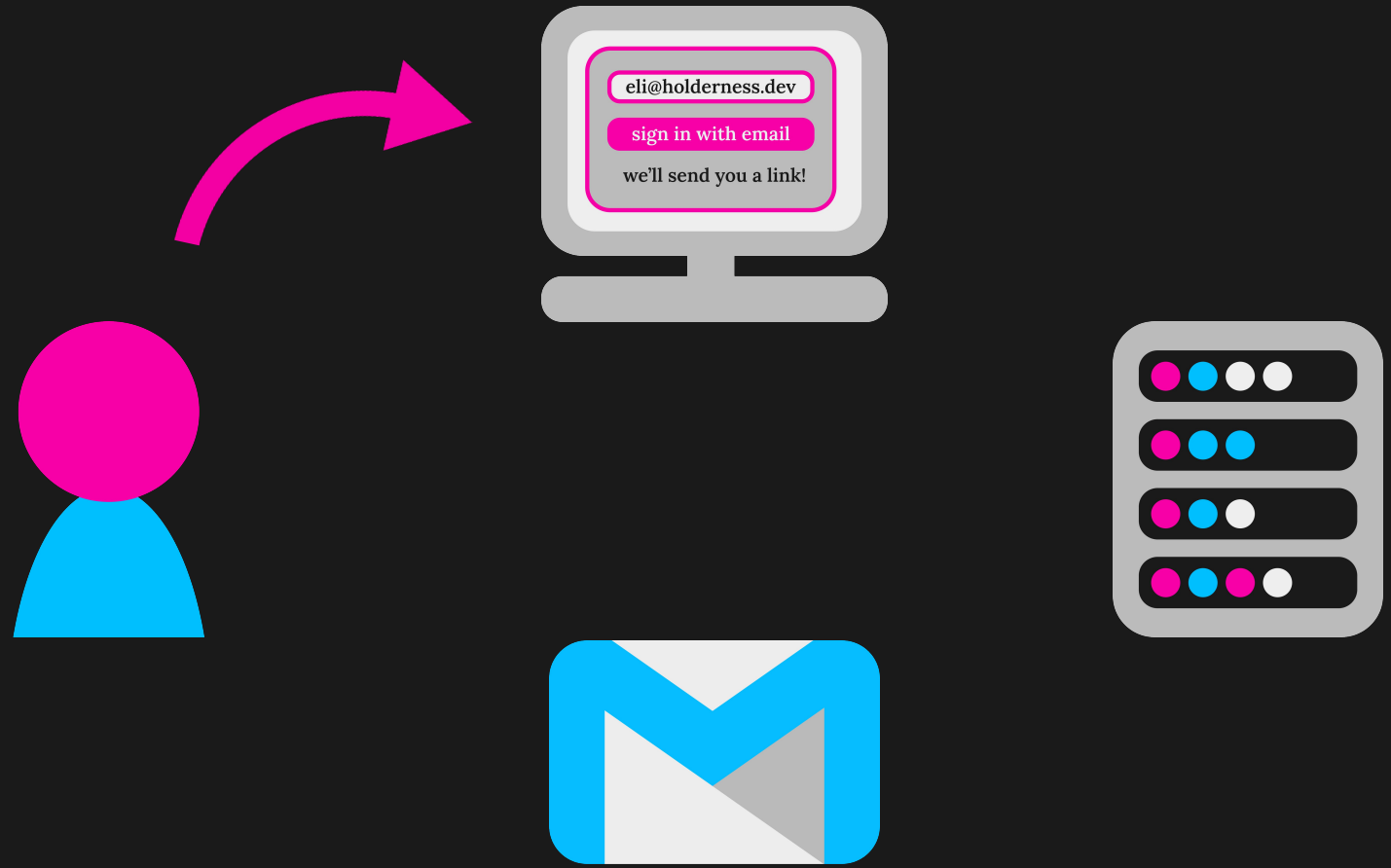
@eli@hachyderm.io

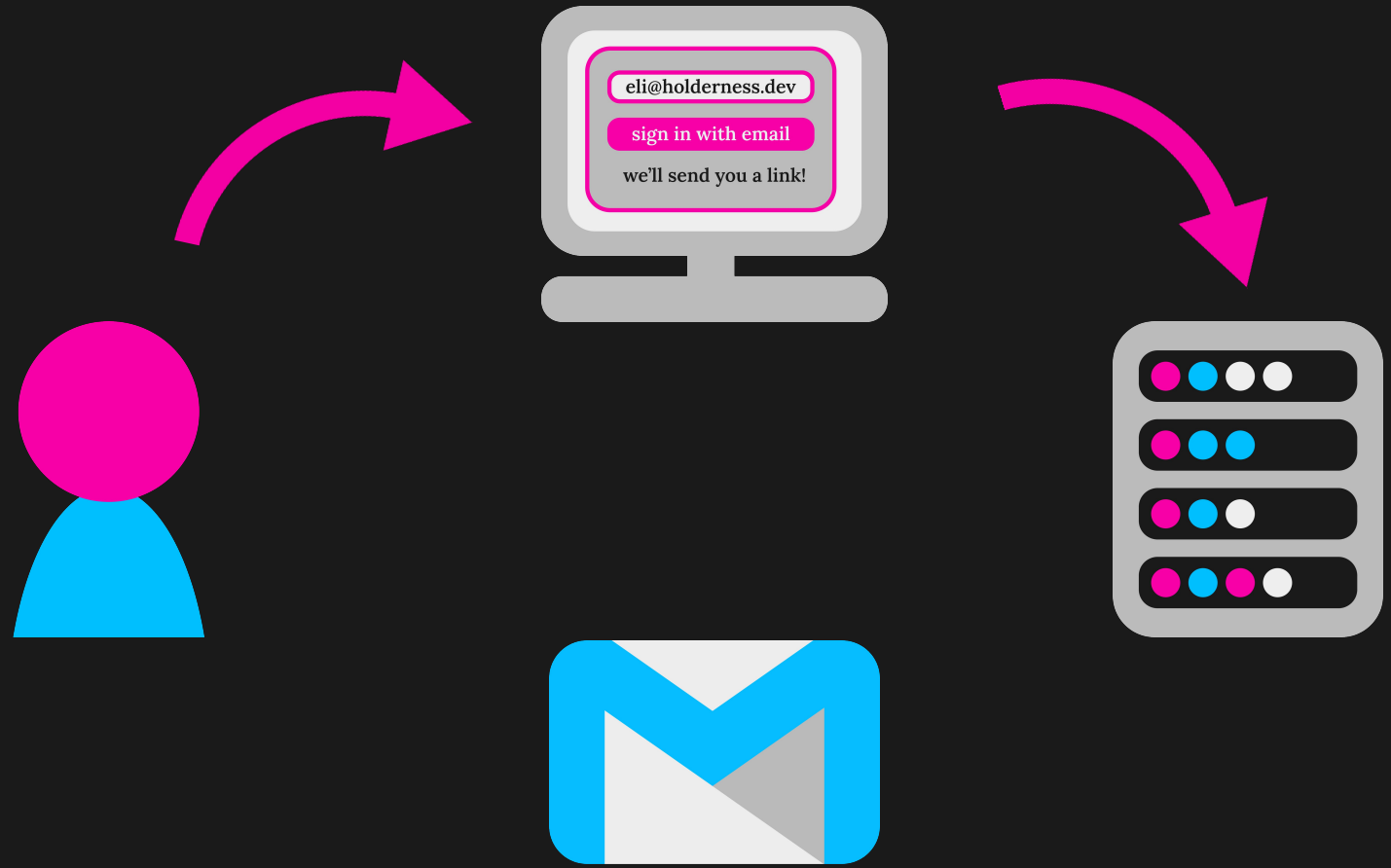

eli@holderness.dev
sign in with email
we'll send you a link!



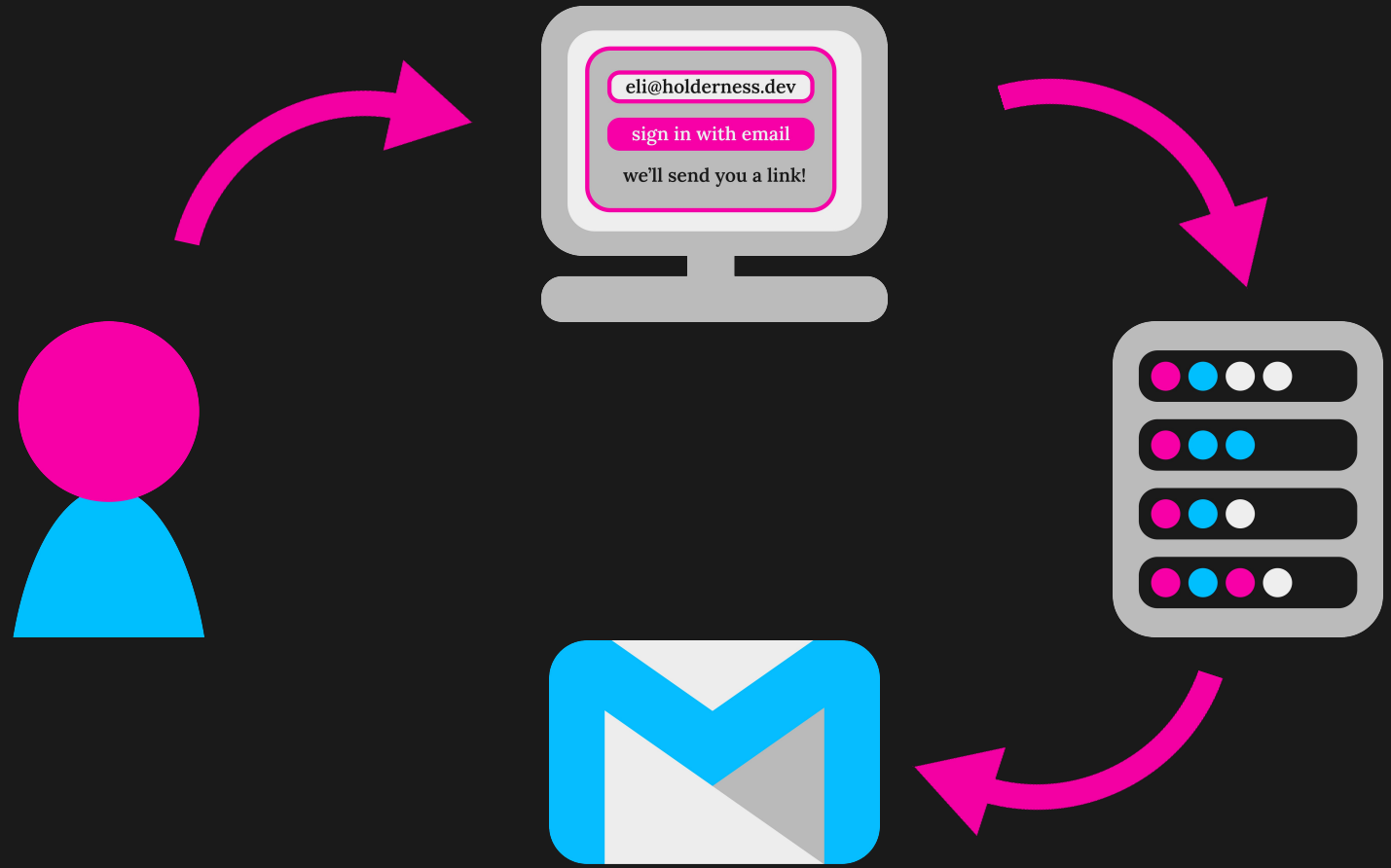


@eli@hachyderm.io

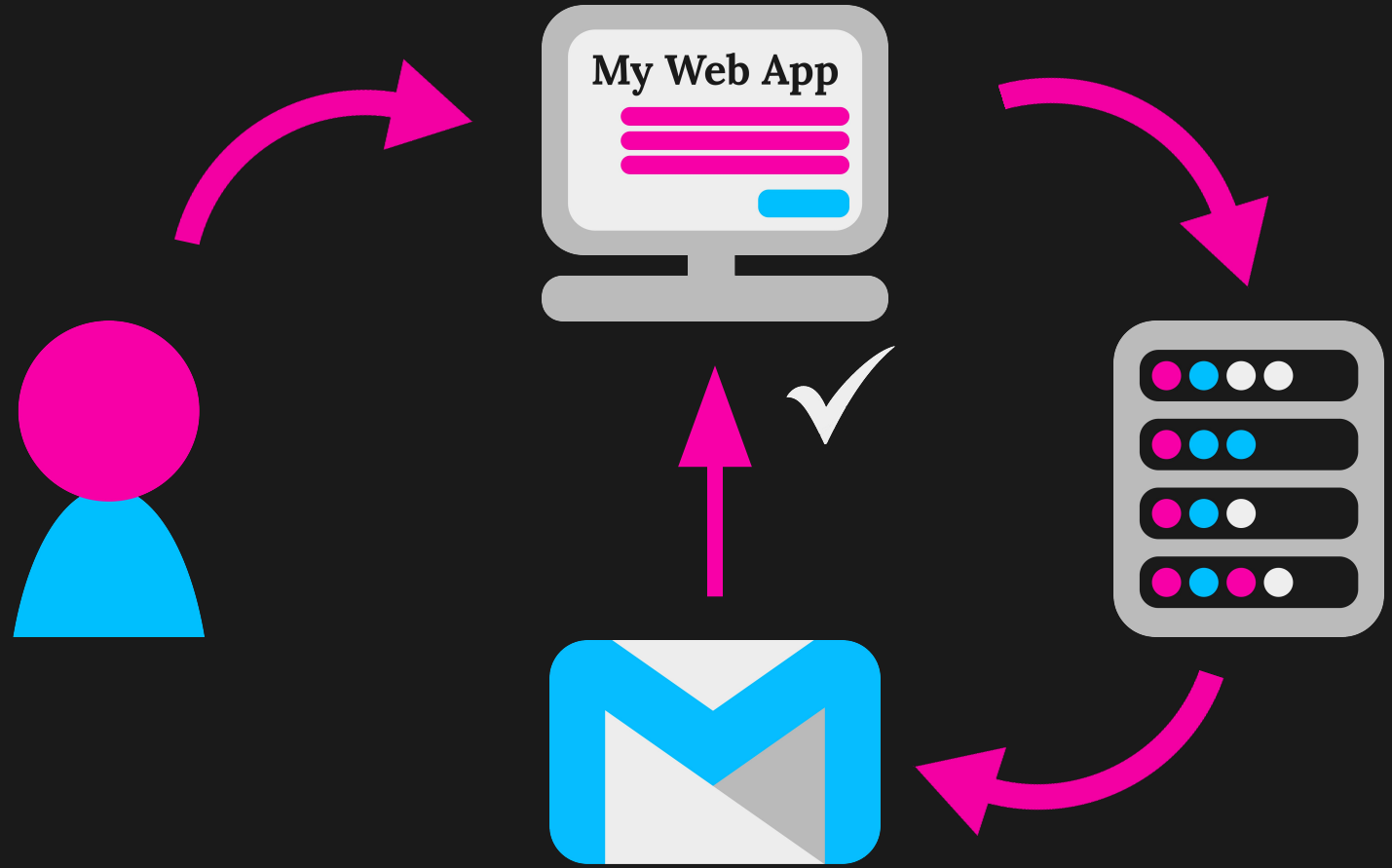

eli@holderness.dev
sign in with email
we'll send you a link!



@eli@hachyderm.io



@eli@hachyderm.io



@eli@hachyderm.io

@eli@hachyderm.io

@eli@hachyderm.io

Now, think about how many online accounts you have...

Now, think about how many online accounts you have...

... and imagine setting up separate two-factor authentication for all of them.

Or, think about how many online accounts an average employee needs...

Or, think about how many online accounts an average employee needs...

... and imagine having to administrate all of them - with MFA!

Single Sign-On

@eli@hachyderm.io

Service Provider

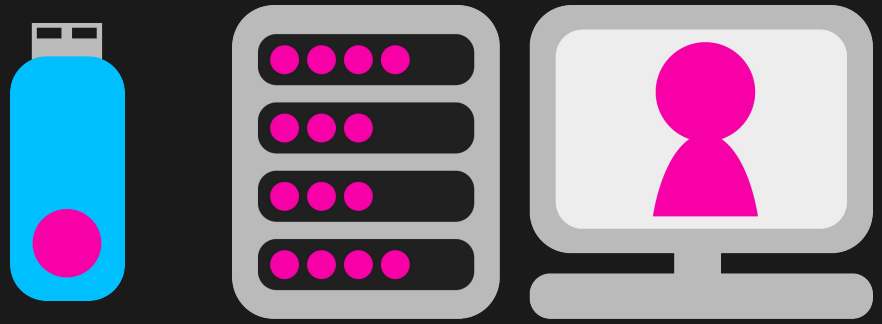
Service Provider

Identity Provider

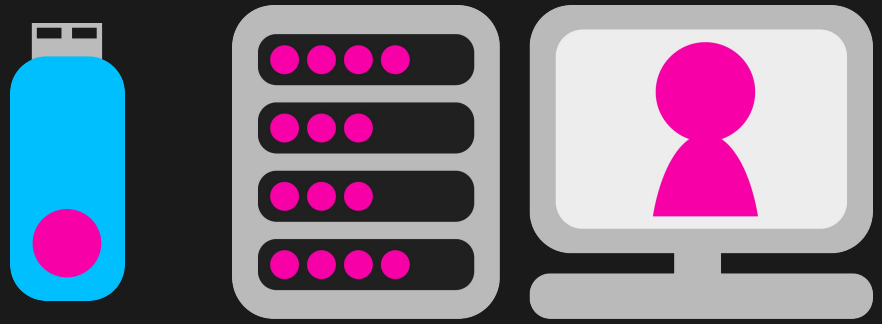
@eli@hachyderm.io

Identity Provider

Identity Provider



Identity Provider

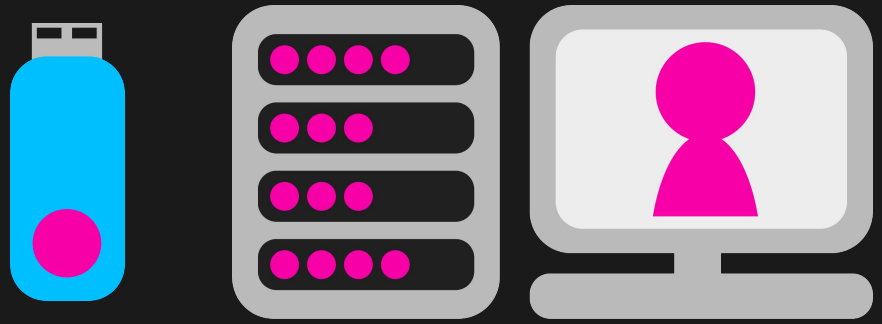


Jane Smith

→

Jane Baker

Identity Provider

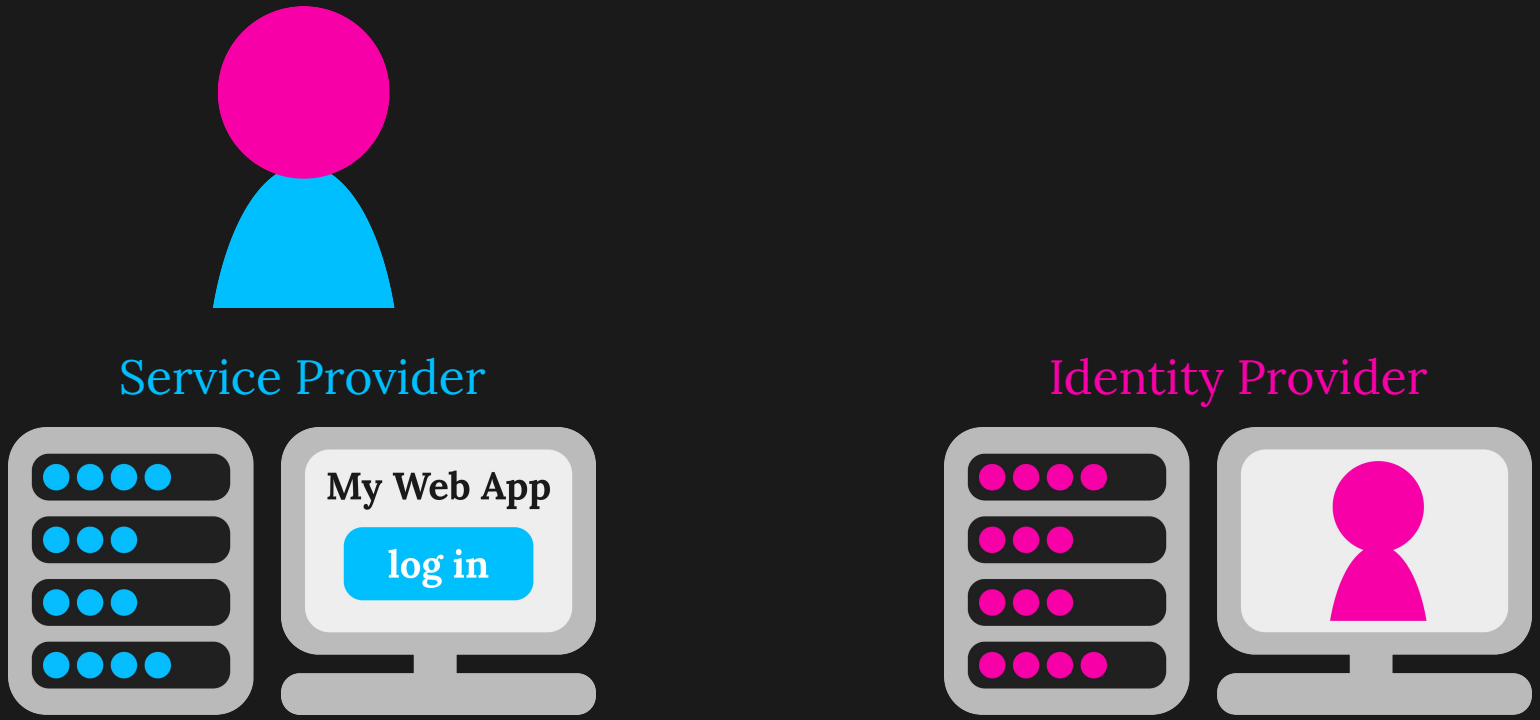


Jane Smith

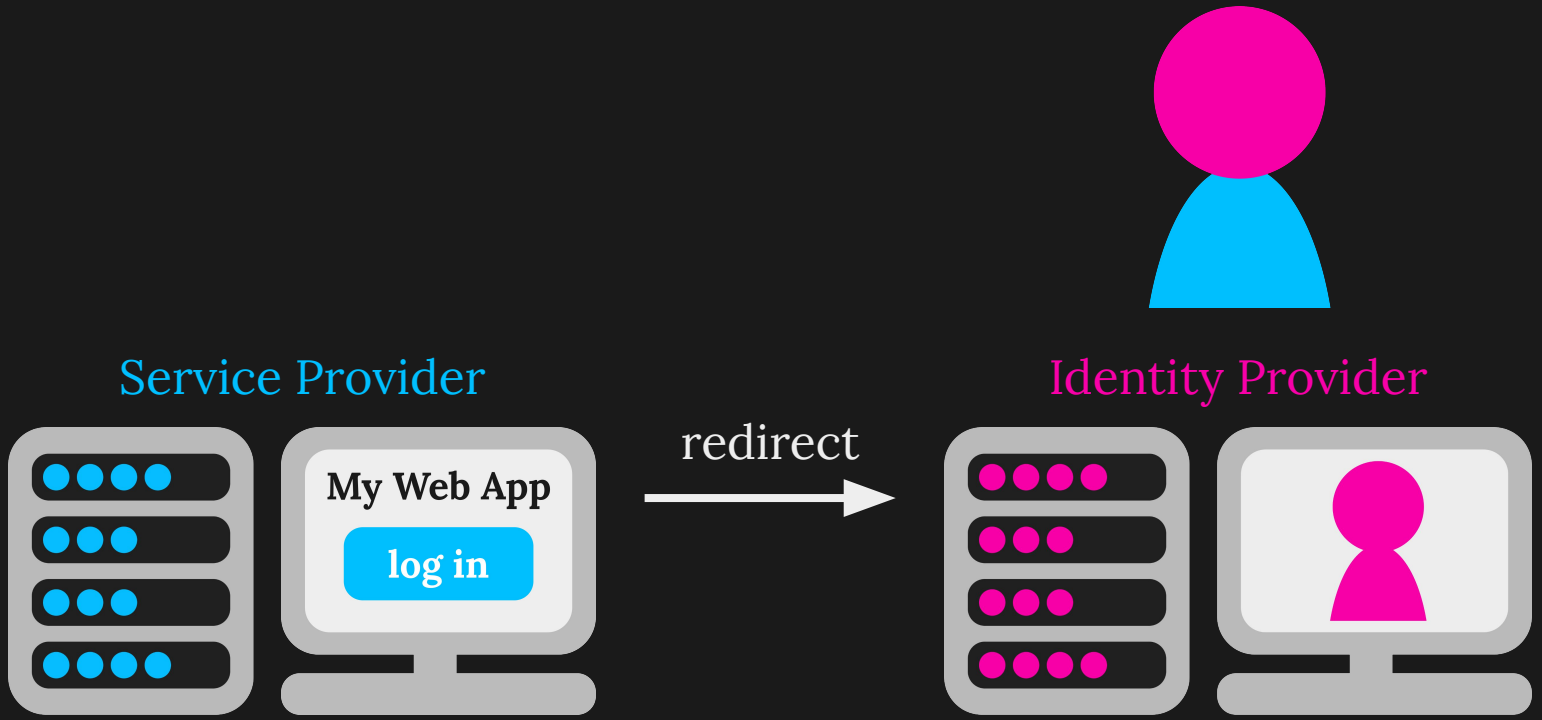
→

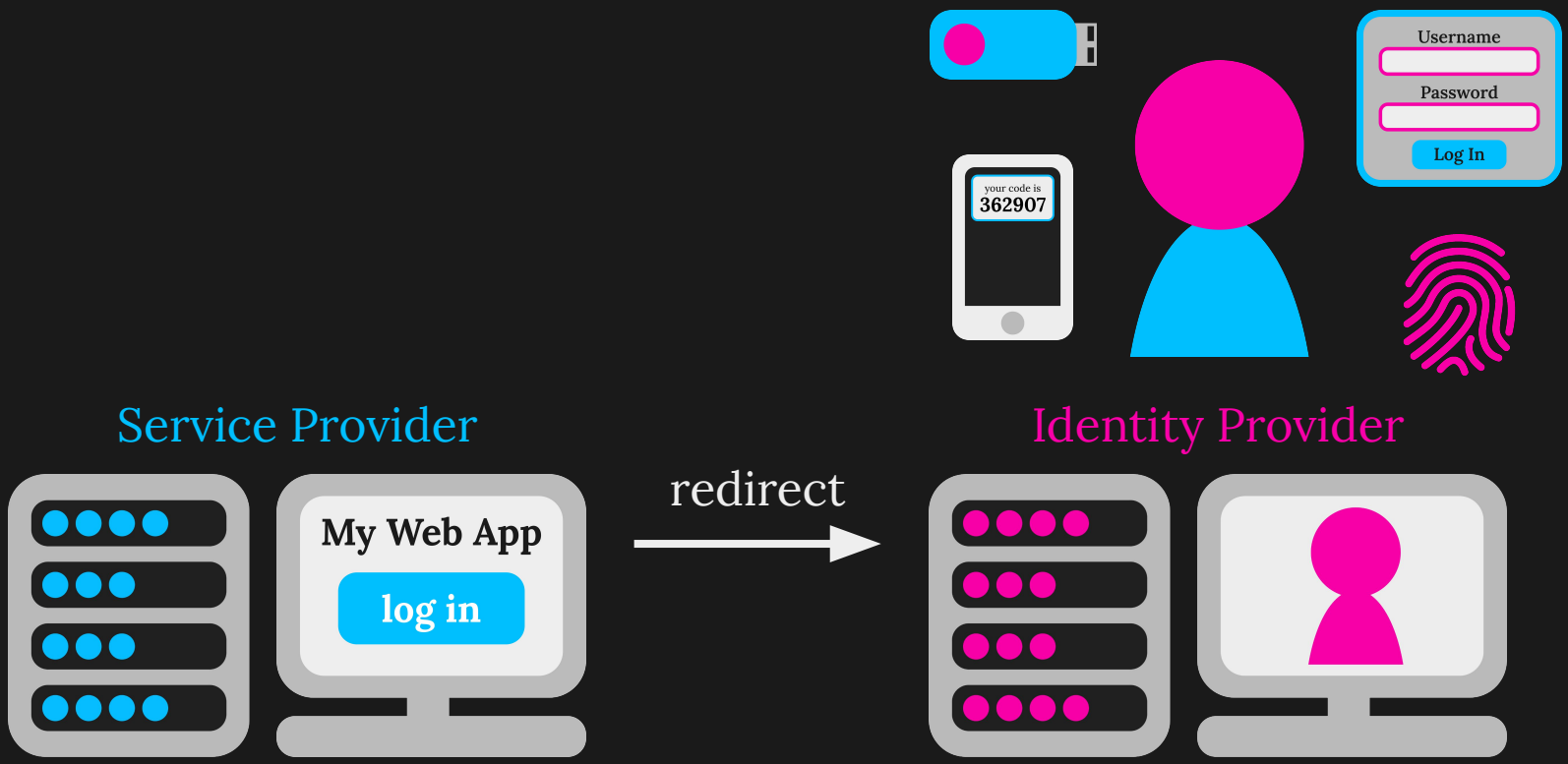
Jane Baker

@eli@hachyderm.io

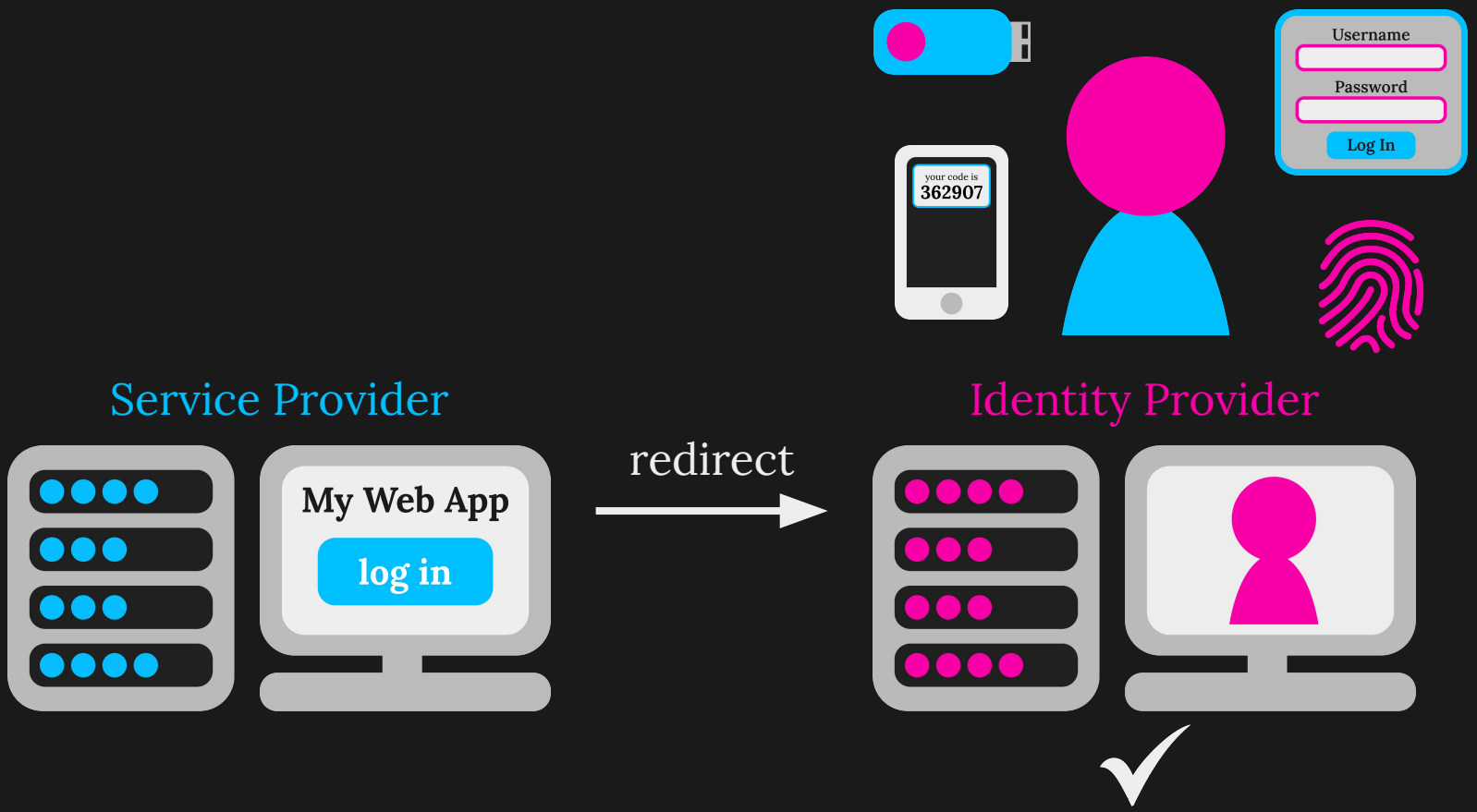


@eli@hachyderm.io

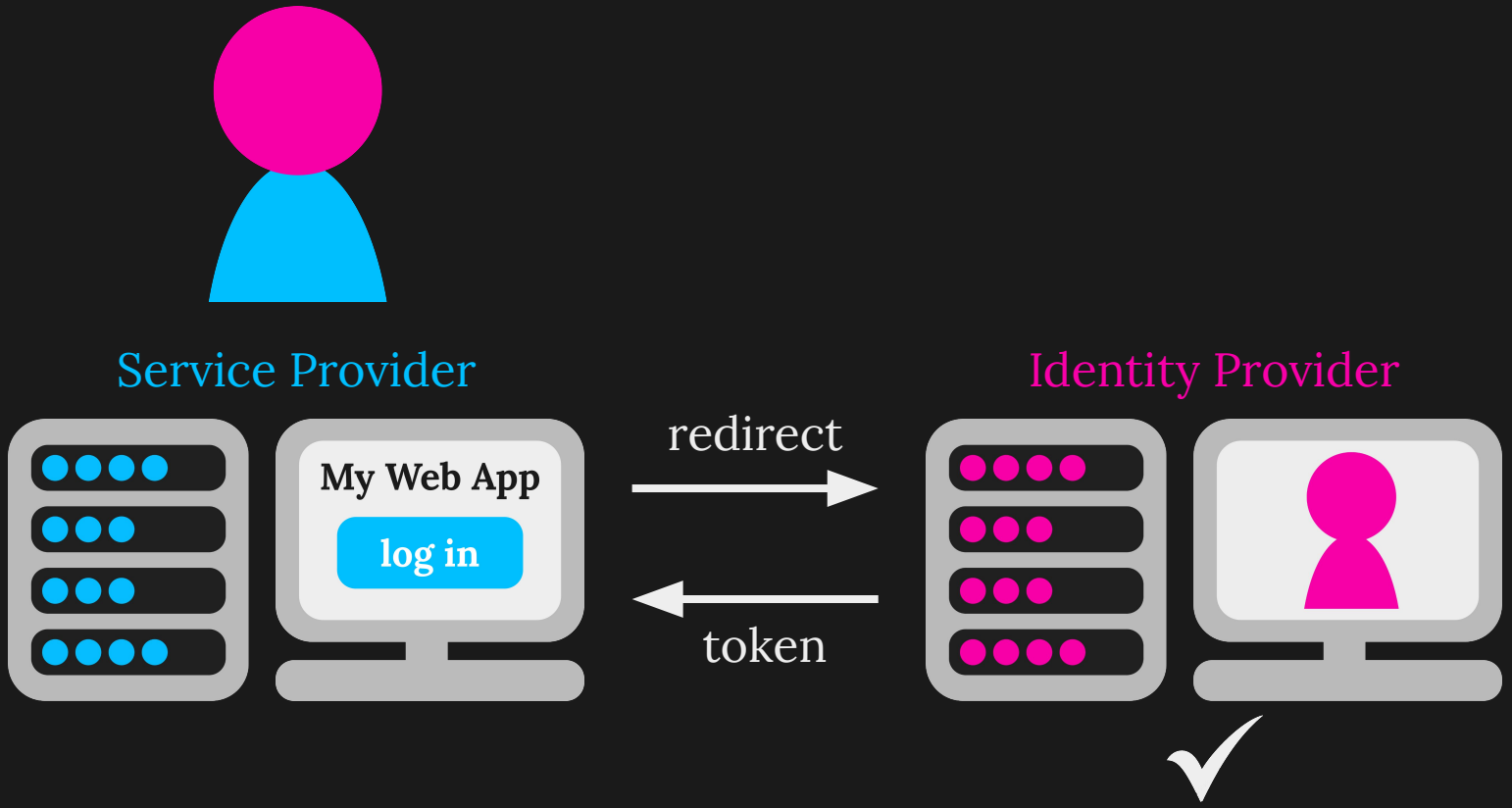

Service Provider
My Web App
log in
redirect
Identity Provider



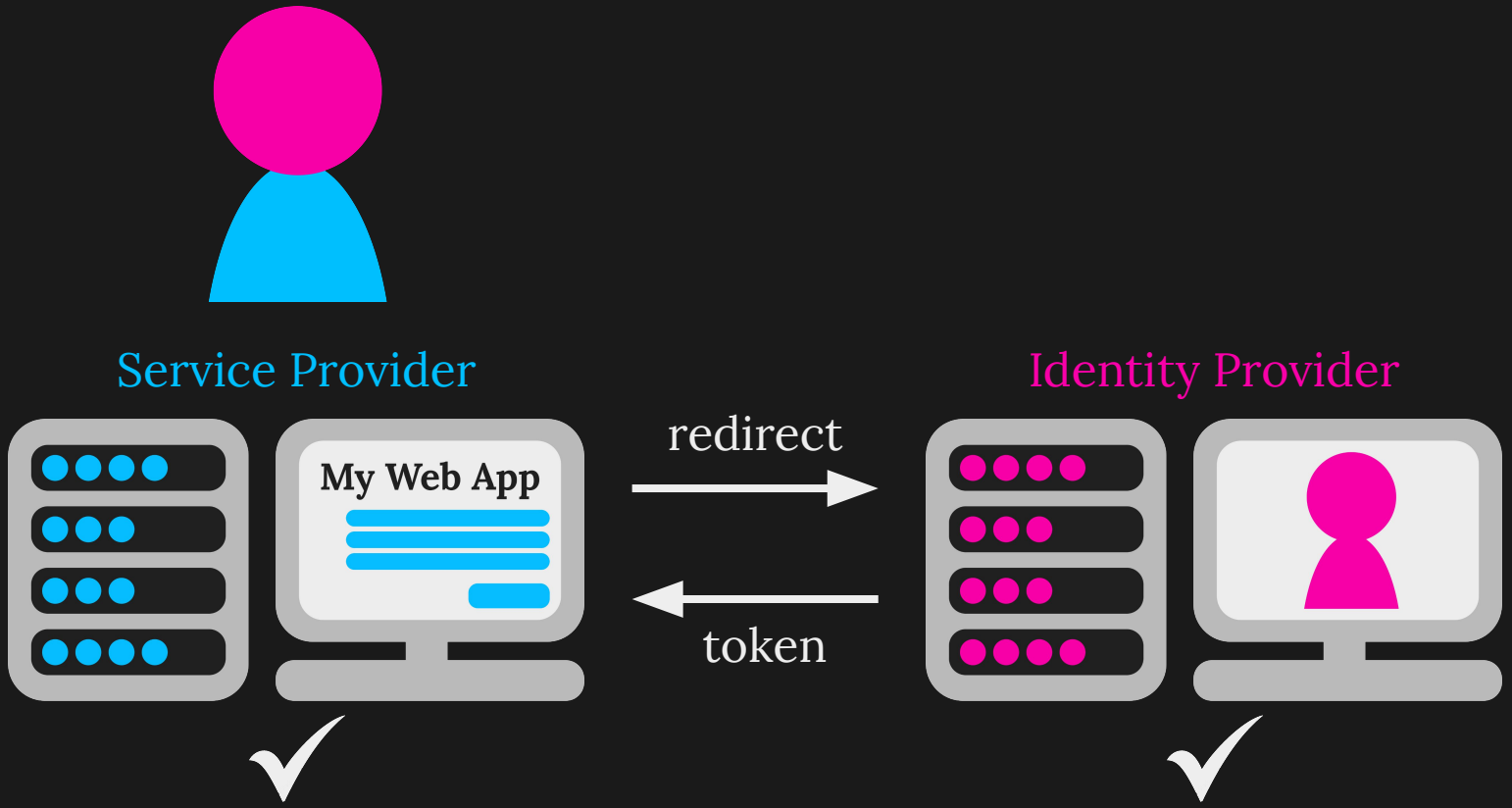
@eli@hachyderm.io



@eli@hachyderm.io



@eli@hachyderm.io



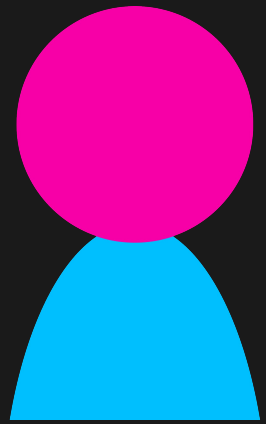
@eli@hachyderm.io

SSO lets you delegate
your authentication to a
third party

web services don't need to store user credentials

@eli@hachyderm.io

users don’t need to
remember lots of passwords

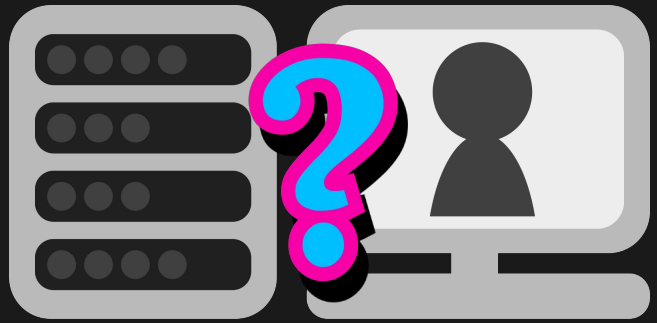


@eli@hachyderm.io

Identity Provider

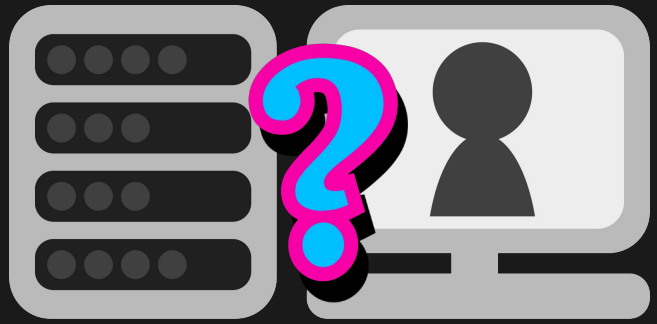
@eli@hachyderm.io

Identity Provider



@eli@hachyderm.io

Identity Provider



@eli@hachyderm.io

How can we implement it?

How can we implement it?

SAML

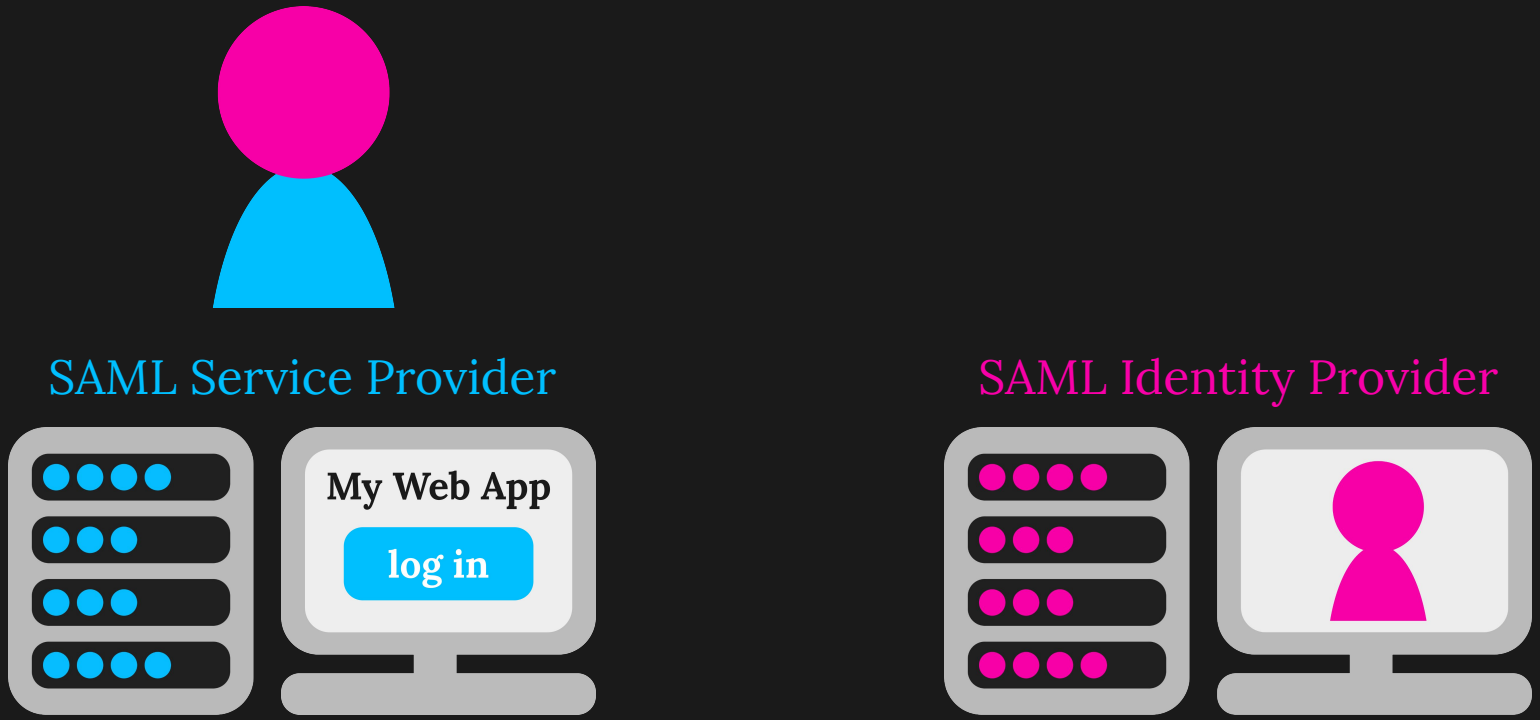
How can we implement it?

SAML

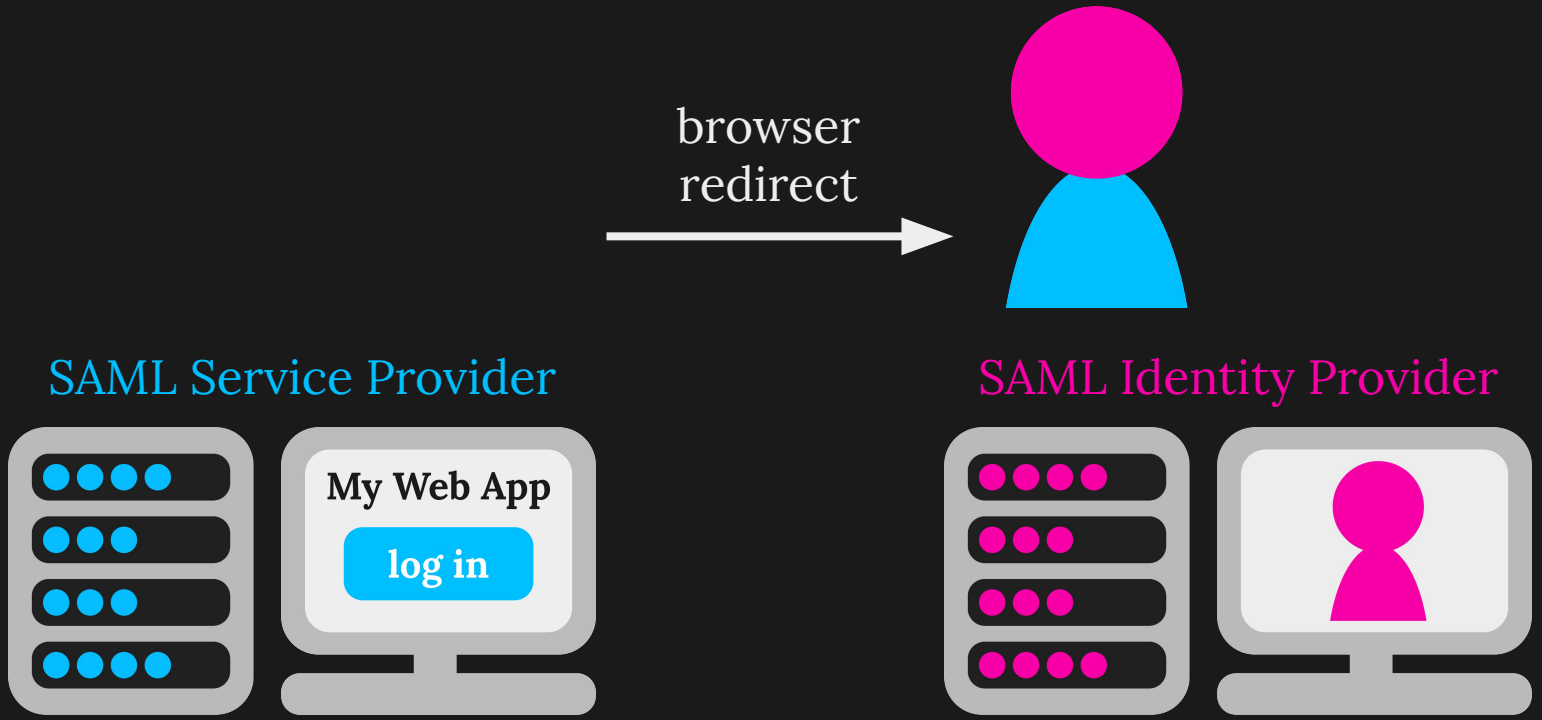
OpenID Connect

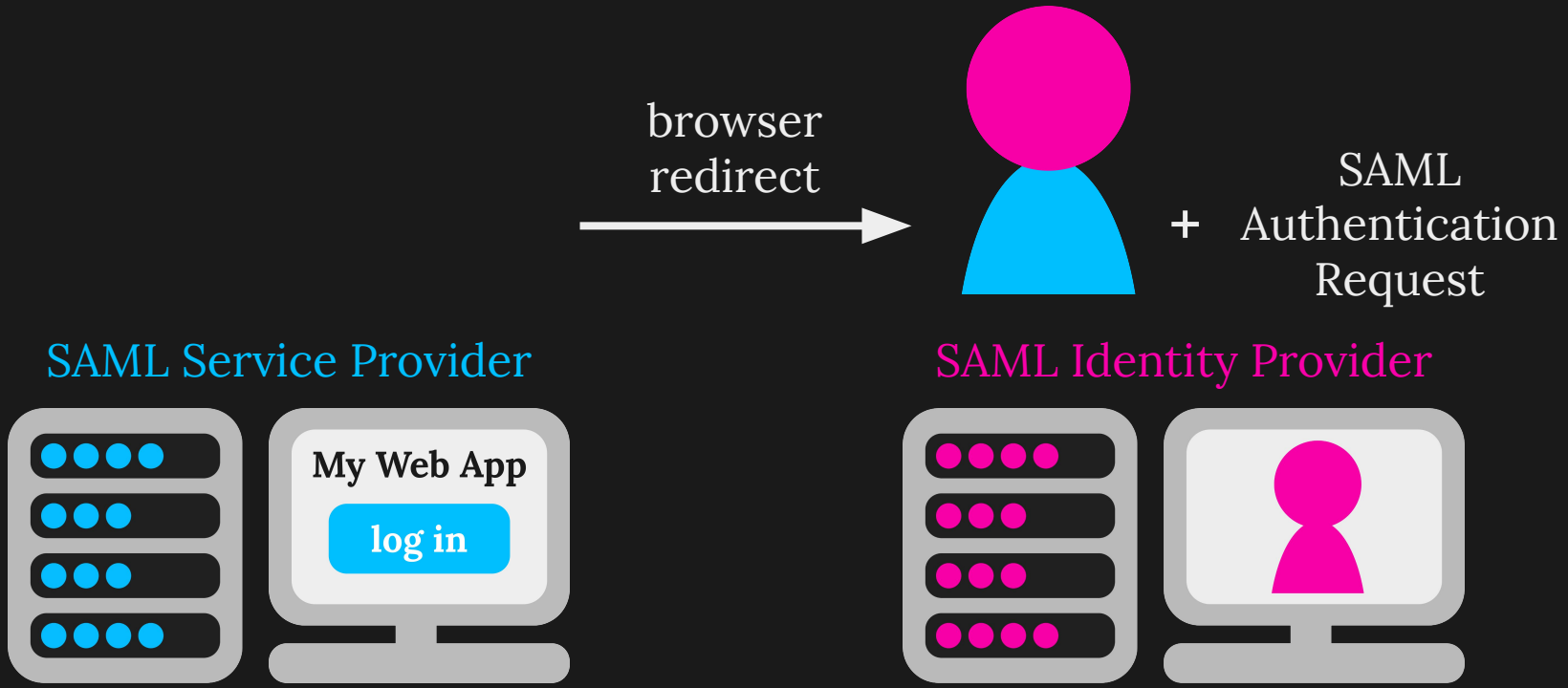
SAML

@eli@hachyderm.io

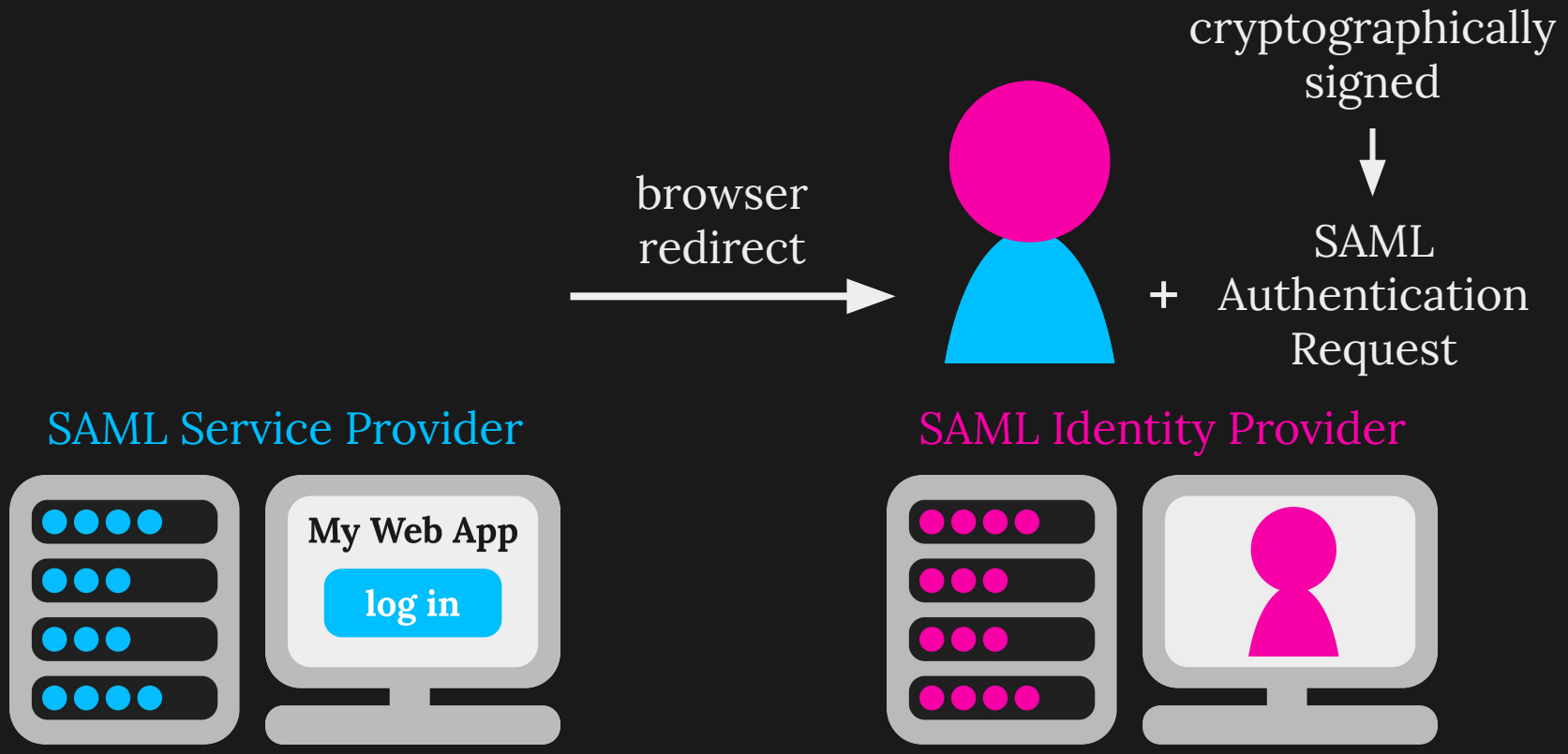


@eli@hachyderm.io

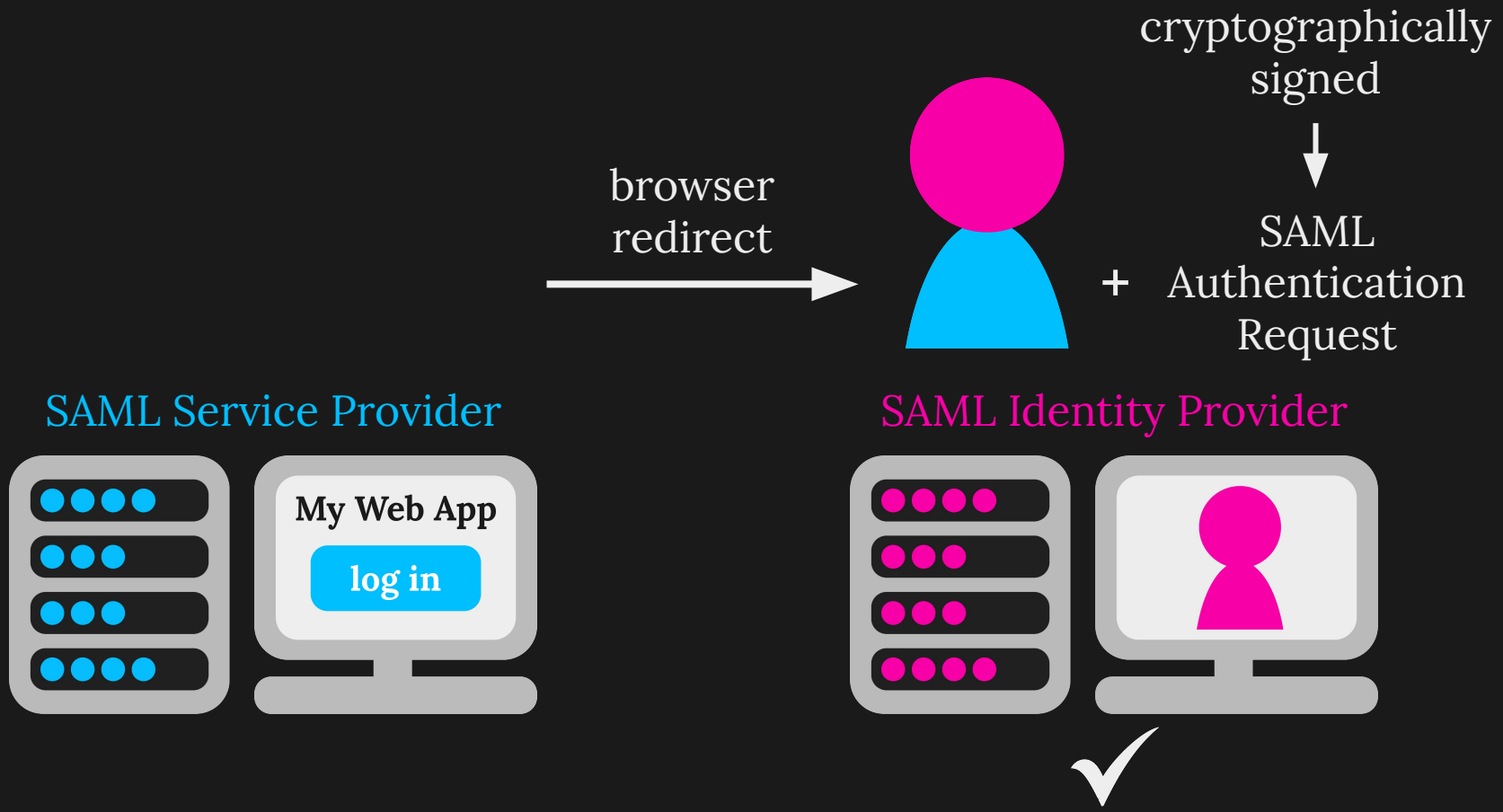

browser
redirect
SAML Service Provider
My Web App
log in
SAML Identity Provider



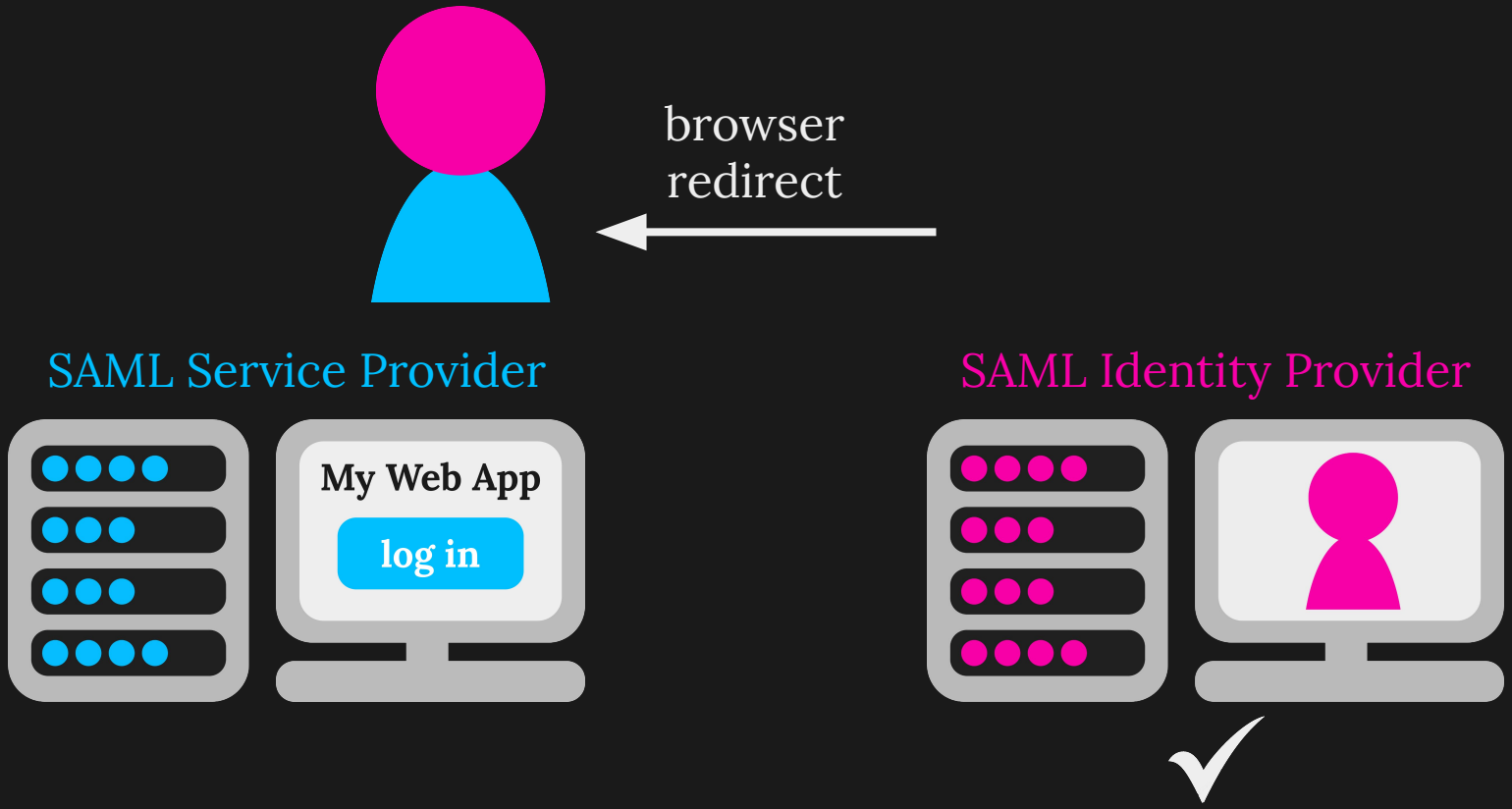
@eli@hachyderm.io



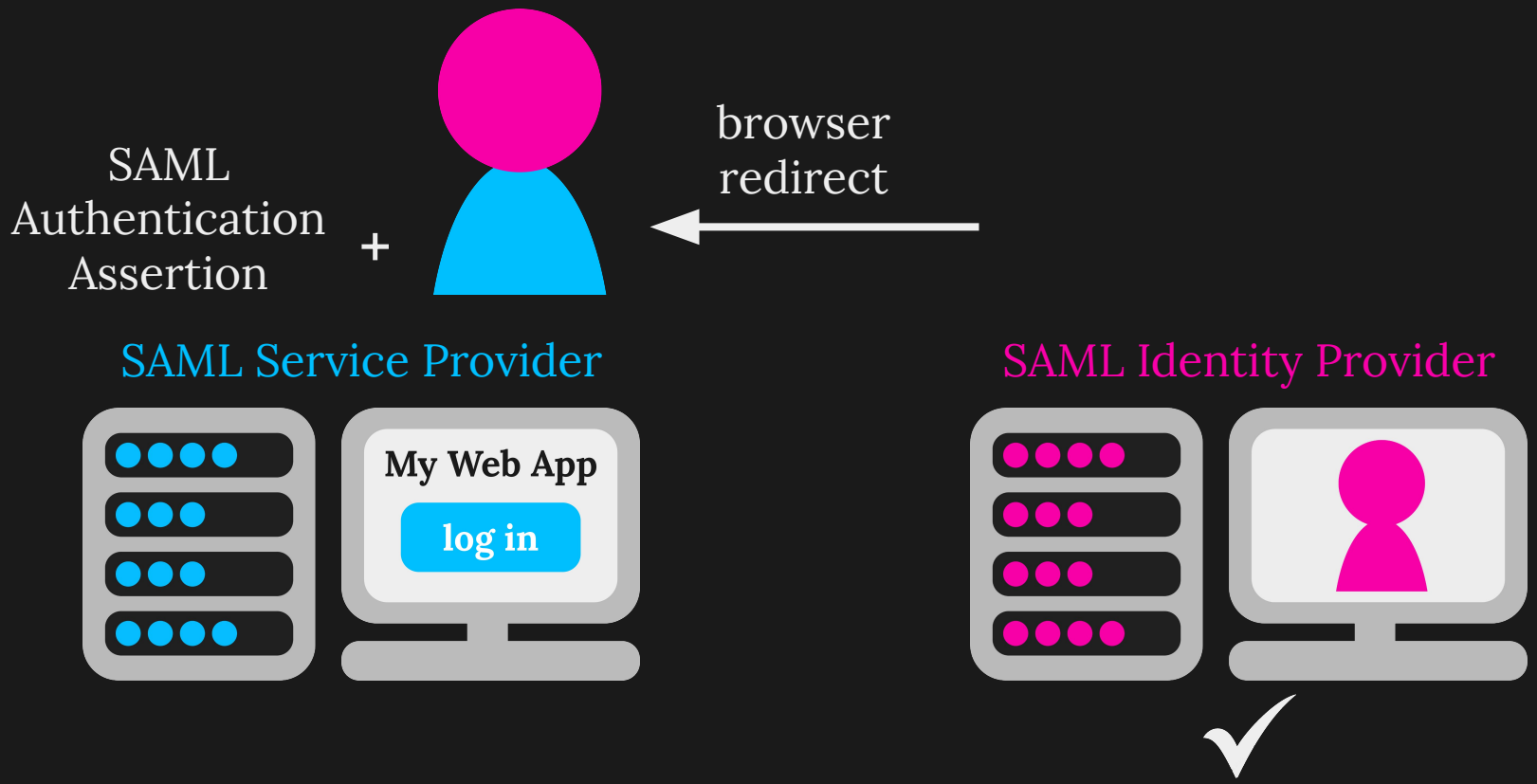
@eli@hachyderm.io



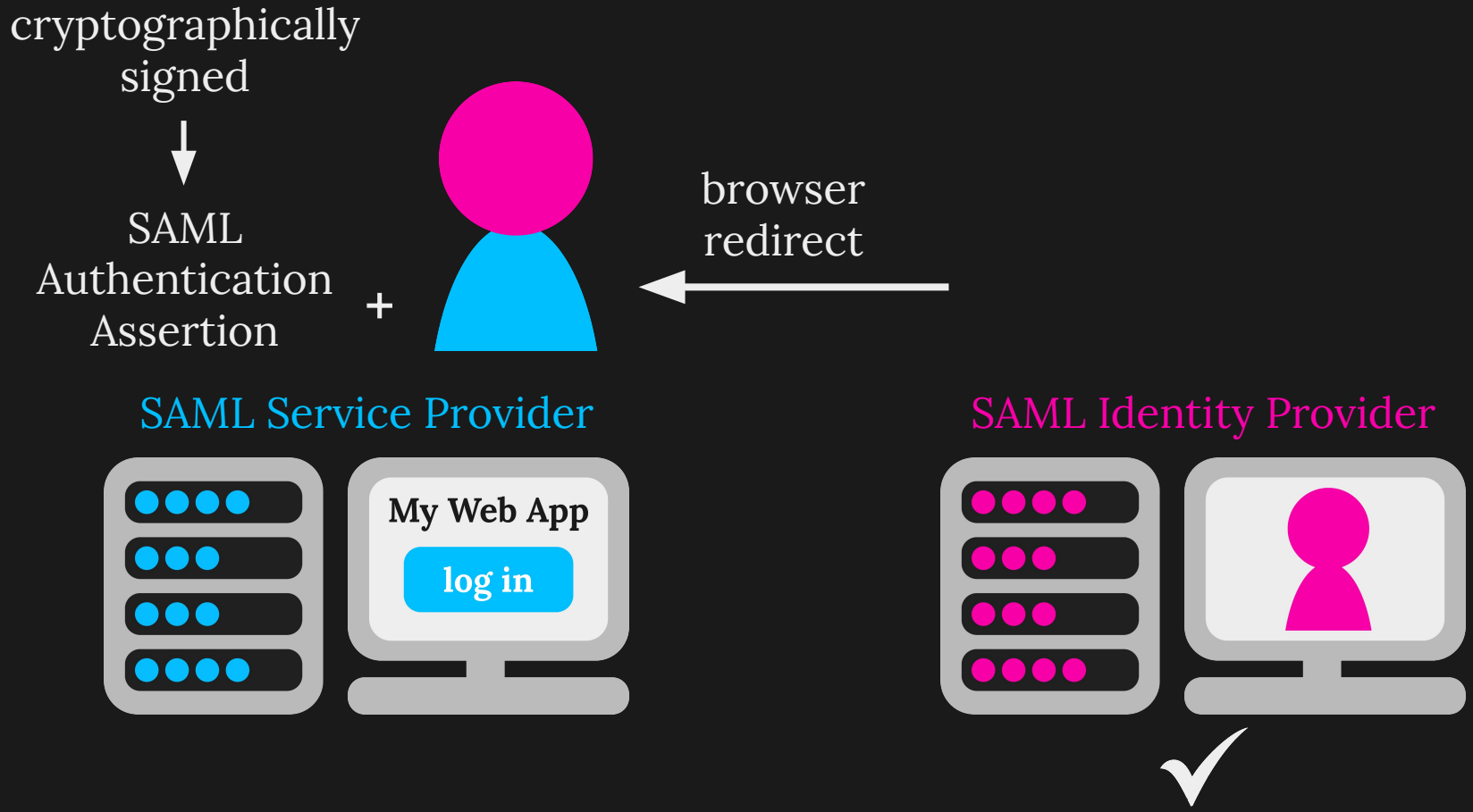
@eli@hachyderm.io



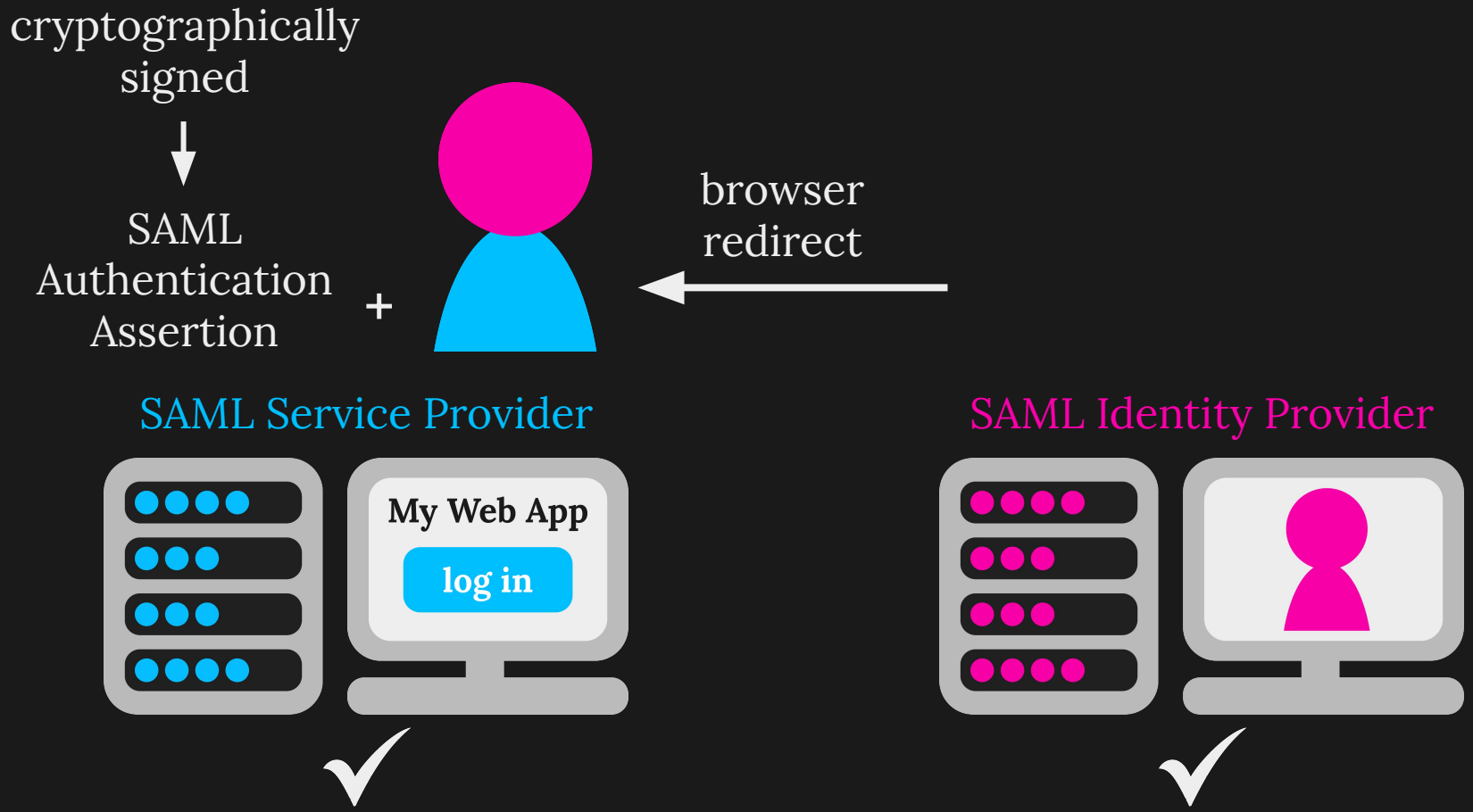
@eli@hachyderm.io



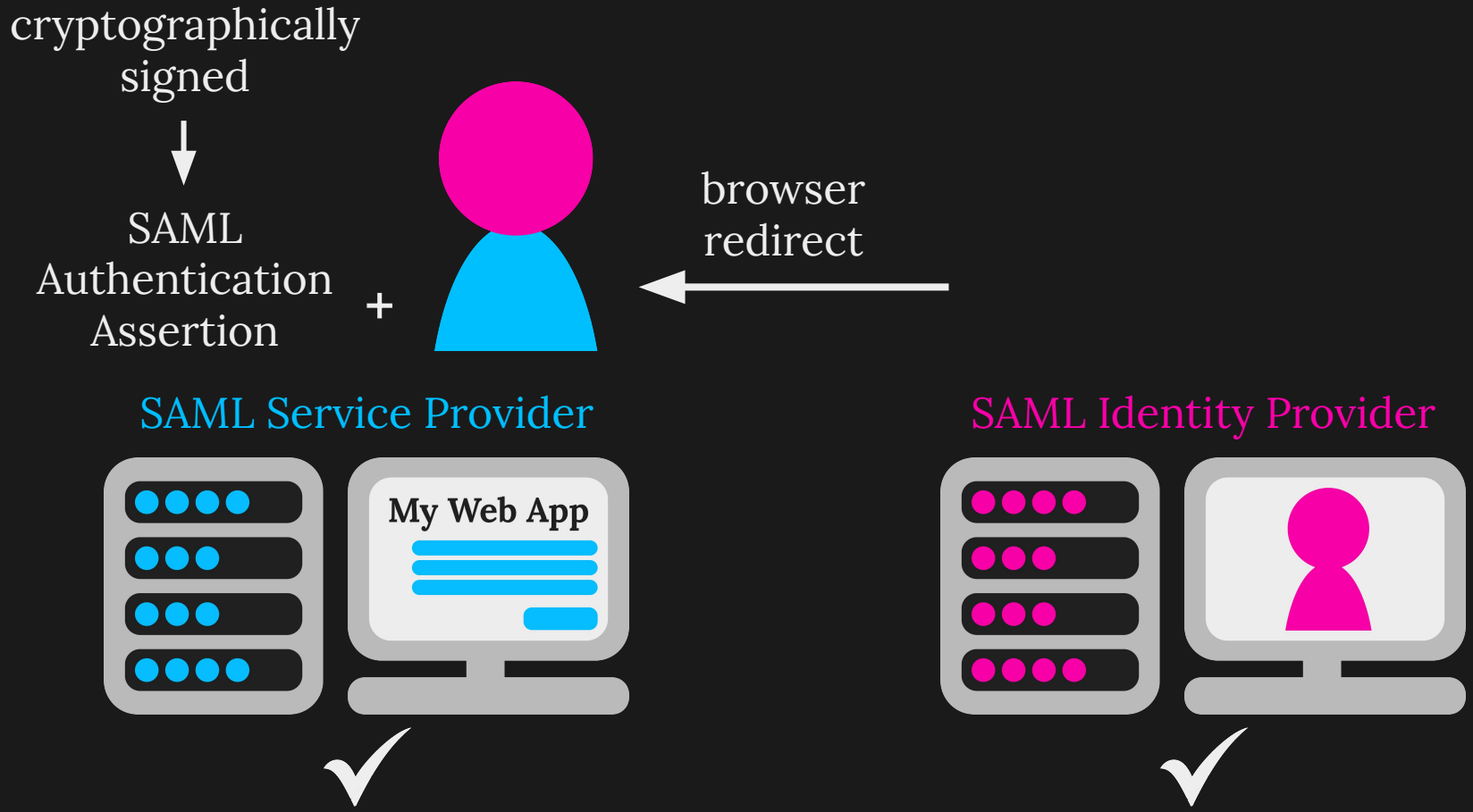
@eli@hachyderm.io



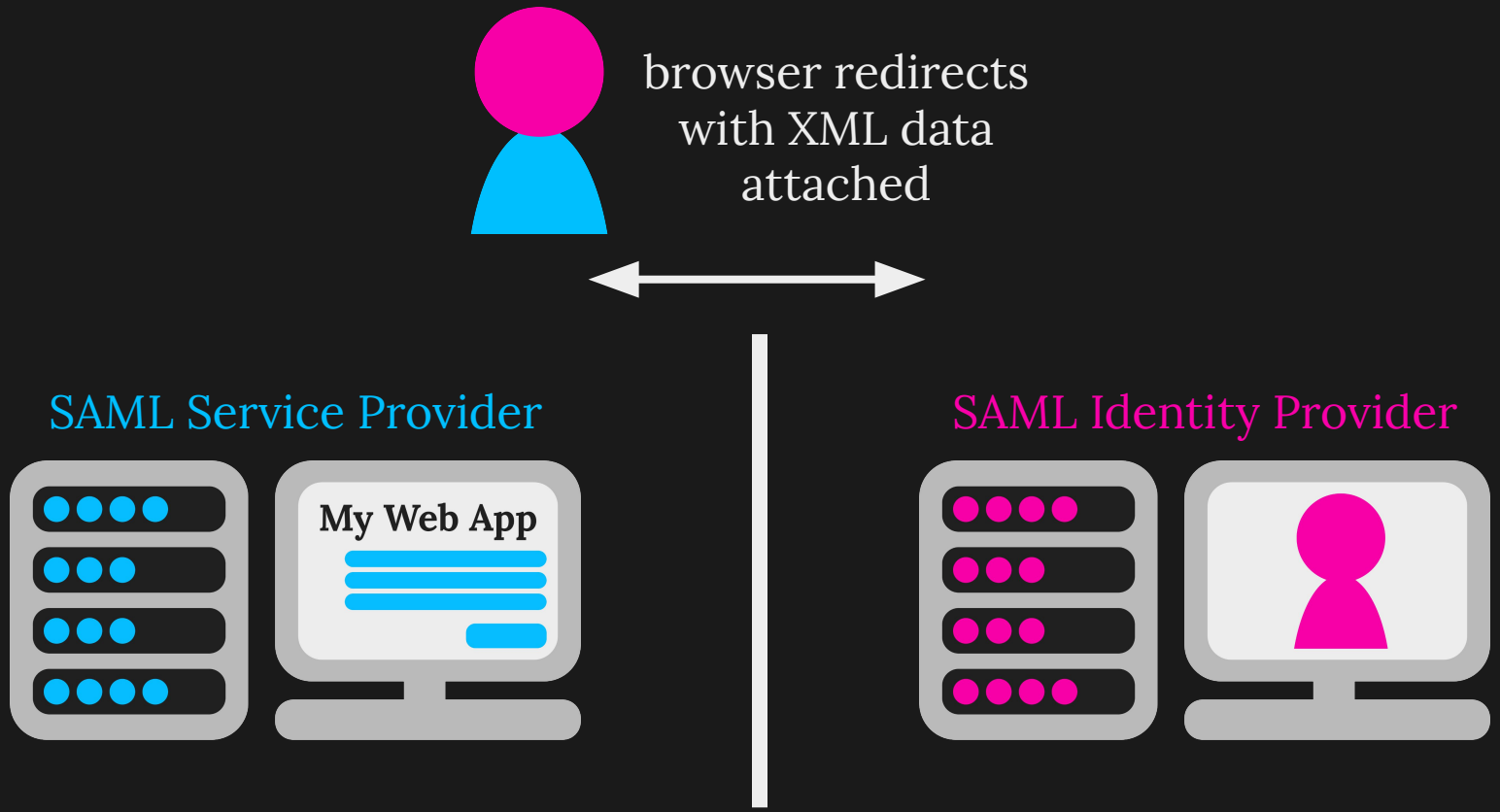
@eli@hachyderm.io



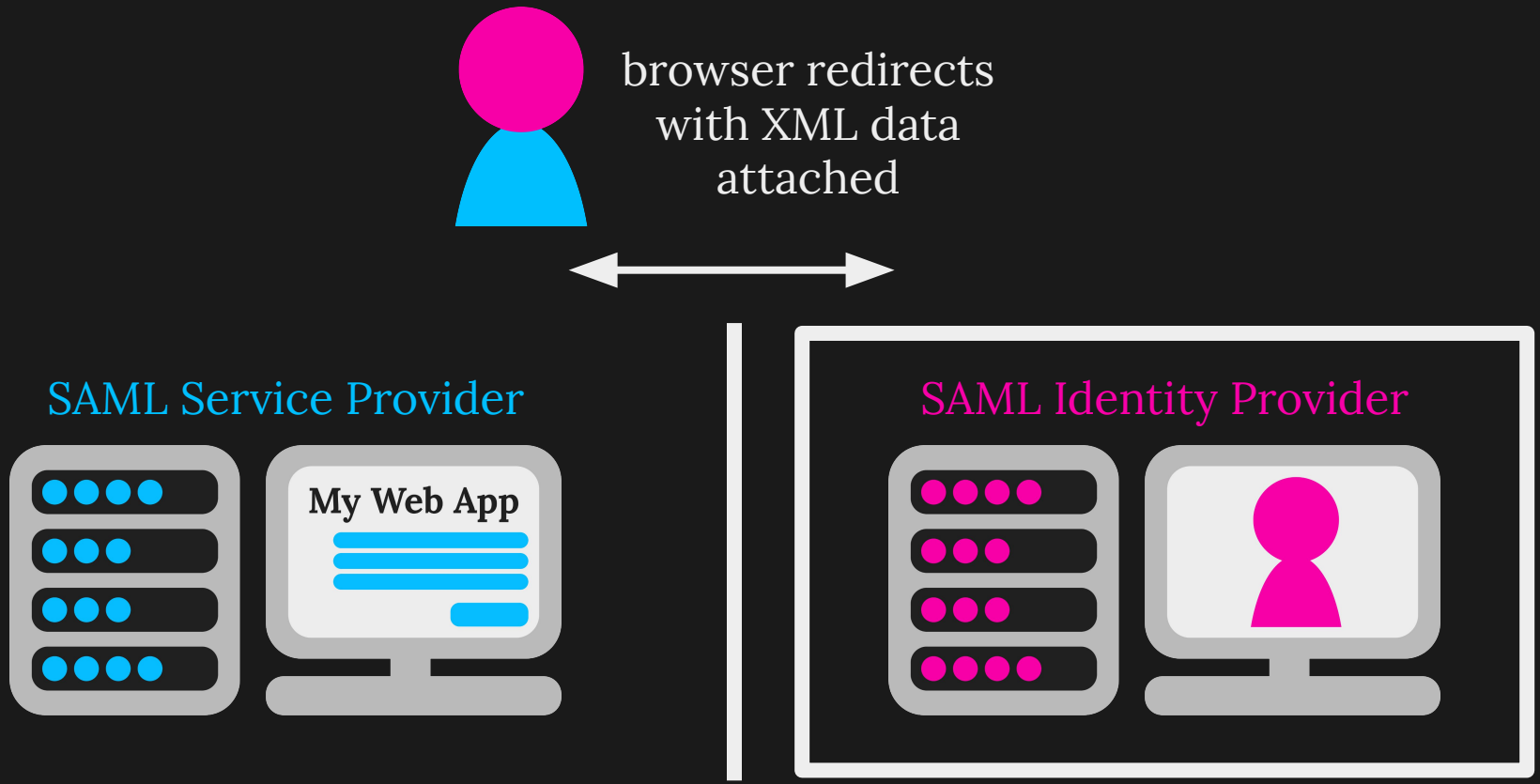
@eli@hachyderm.io



@eli@hachyderm.io



@eli@hachyderm.io



@eli@hachyderm.io

Signing XML data is hard.

Really hard.

@eli@hachyderm.io

<saml:Issuer> ... </saml:Issuer>

< saml:Issuer > ... < / saml:Issuer >

@eli@hachyderm.io

<saml:Issuer> … </saml:Issuer>

=

“The Issuer is …”

=

< saml:Issuer > … < / saml:Issuer >

@eli@hachyderm.io

<saml:Issuer>
...
</saml:Issuer>

< saml:Issuer >
...
< / saml:Issuer >

@eli@hachyderm.io

canonicalisation

<saml:Issuer>
...
</saml:Issuer>

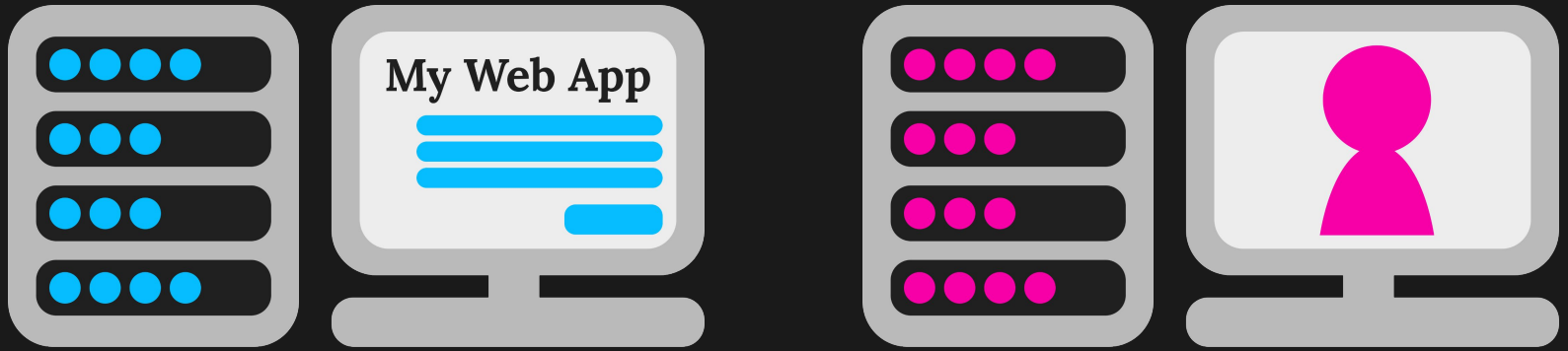
< saml:Issuer >
...
< / saml:Issuer >

“The saml:Issuer is …”

@eli@hachyderm.io

SAML Service Provider

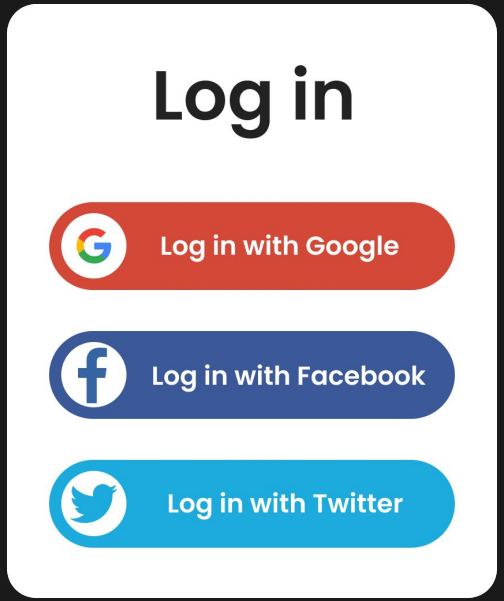
SAML Identity Provider



@eli@hachyderm.io

OpenID Connect

@eli@hachyderm.io



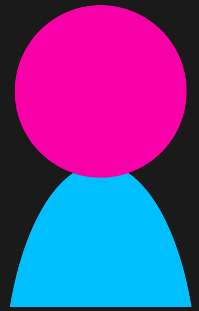
@eli@hachyderm.io

OIDC uses JWTs rather than XML for tokens

@eli@hachyderm.io

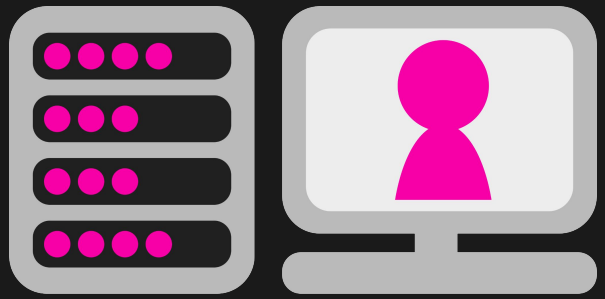
OIDC uses JWTs rather than XML for tokens

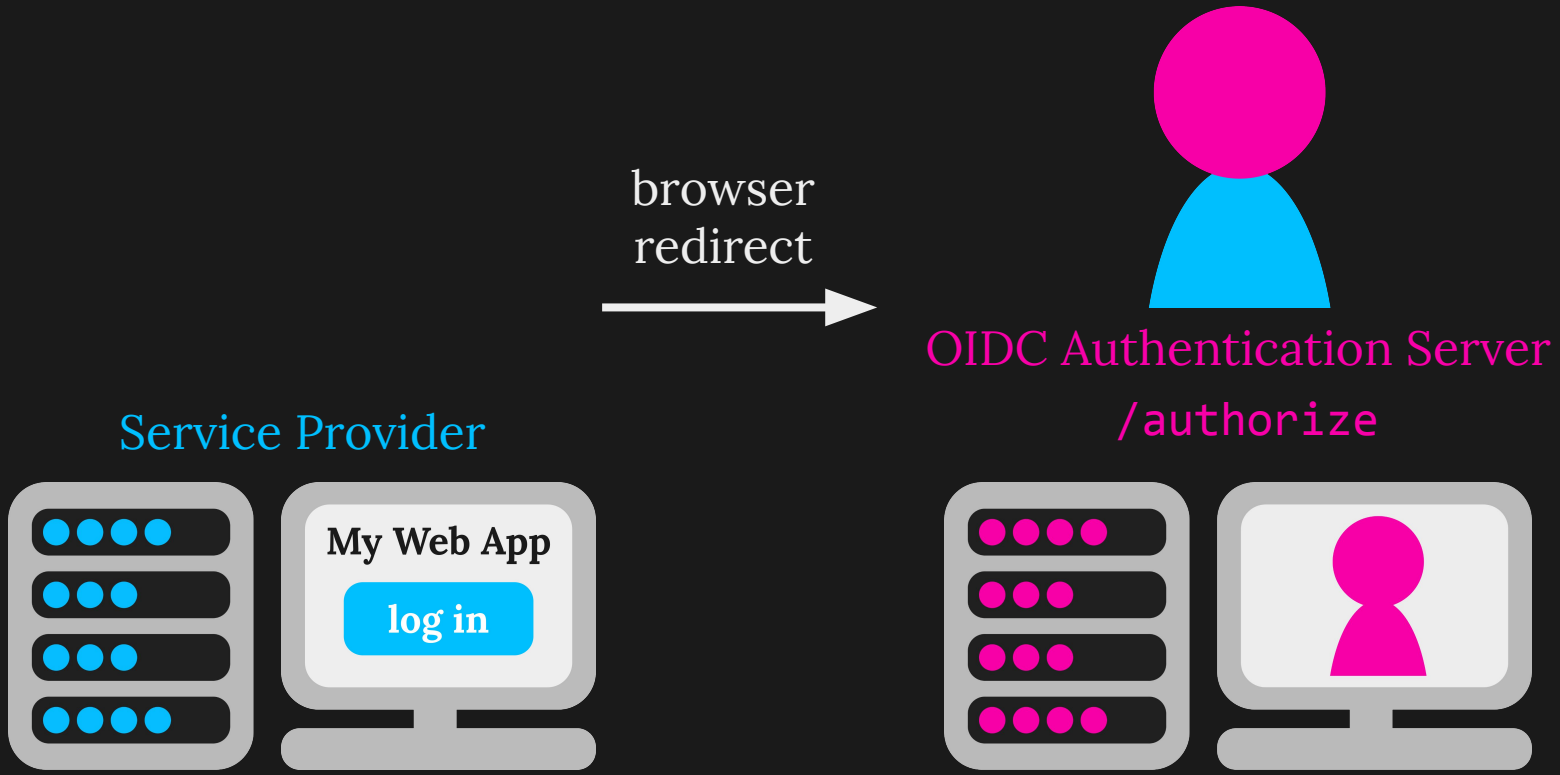
tokens are parsed by the Service Provider, but verified by the Identity Provider

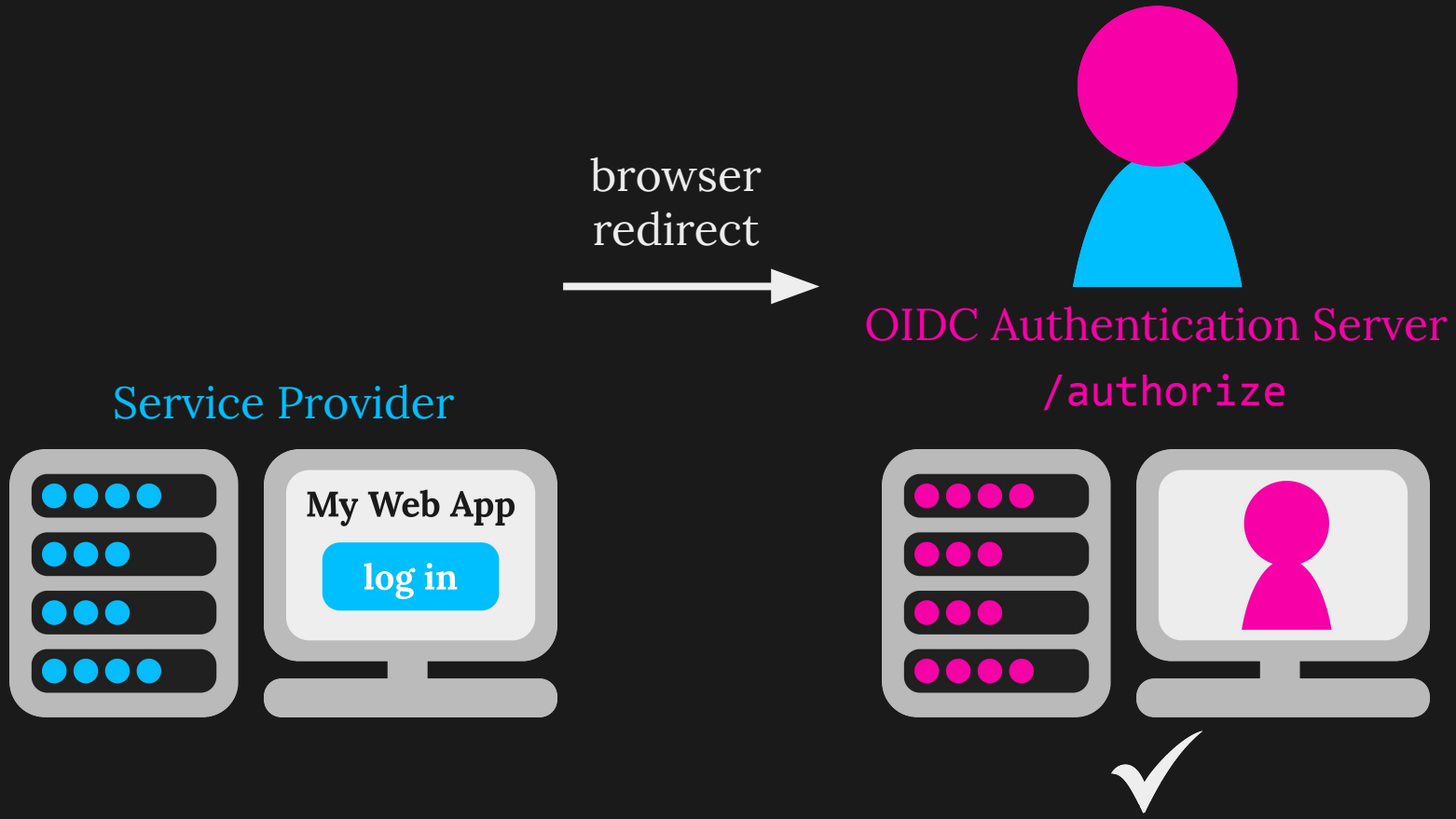


Service Provider

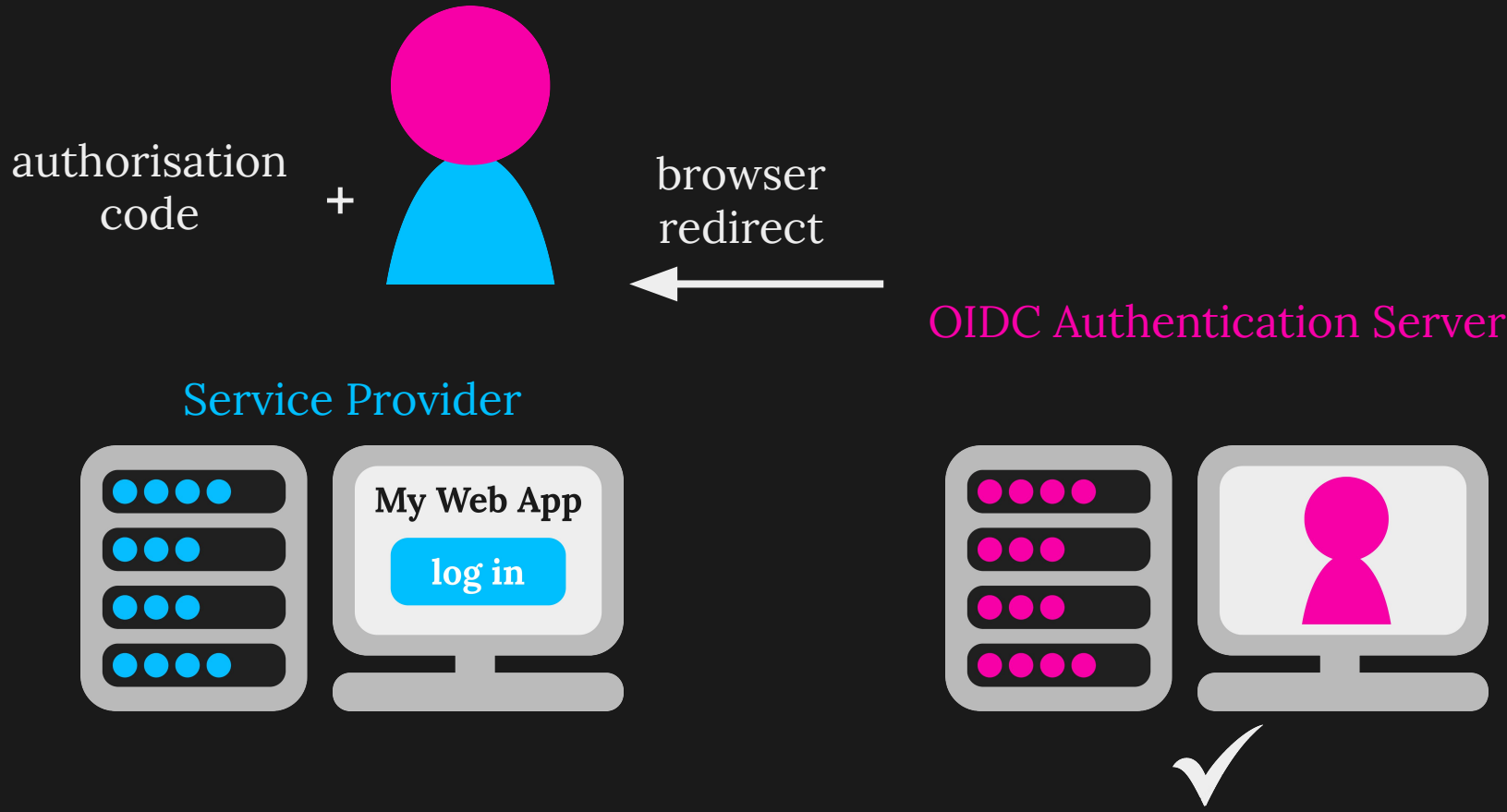
OIDC Authentication Server



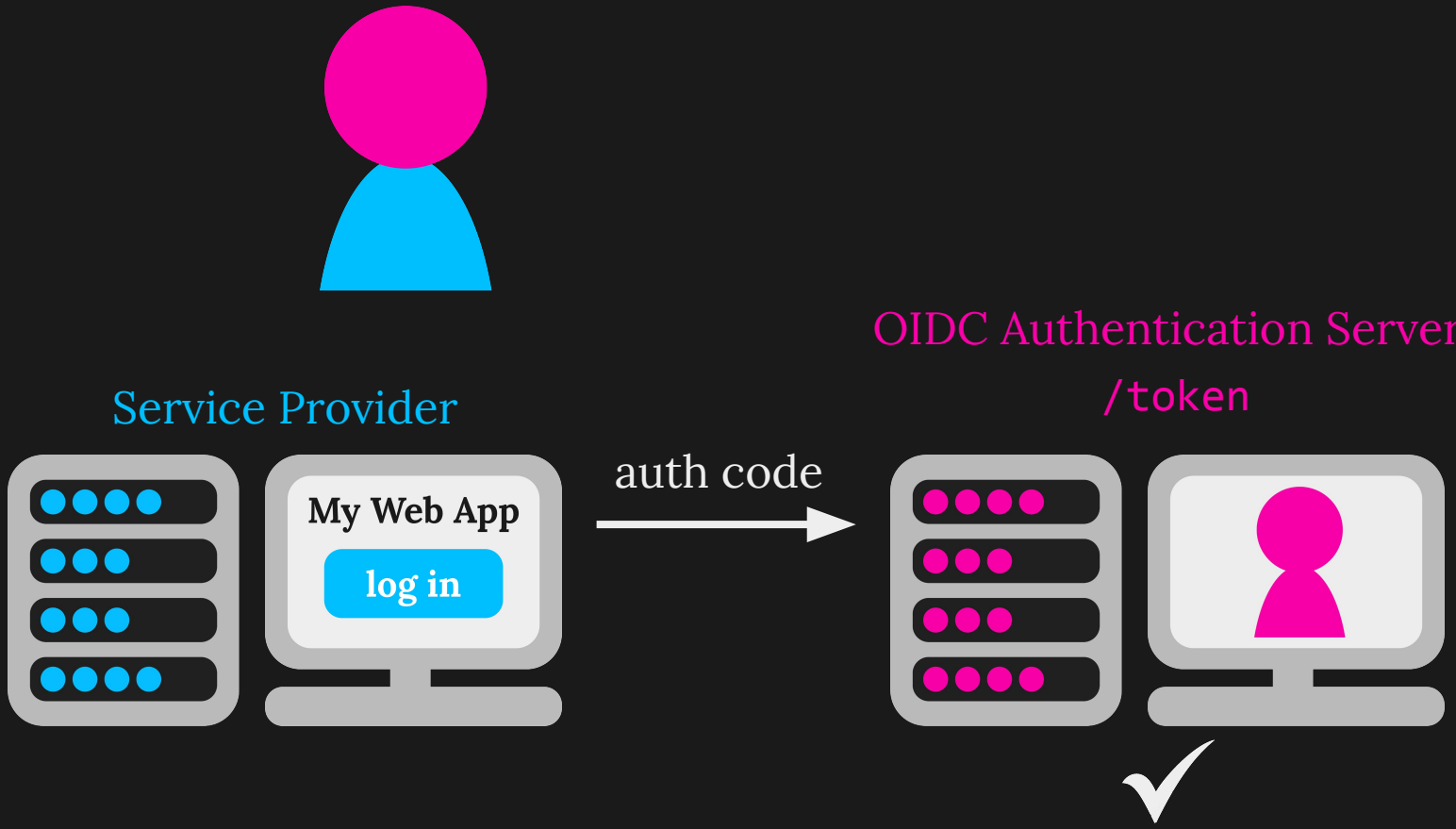

browser
redirect
OIDC Authentication Server
/authorize
Service Provider
My Web App
log in



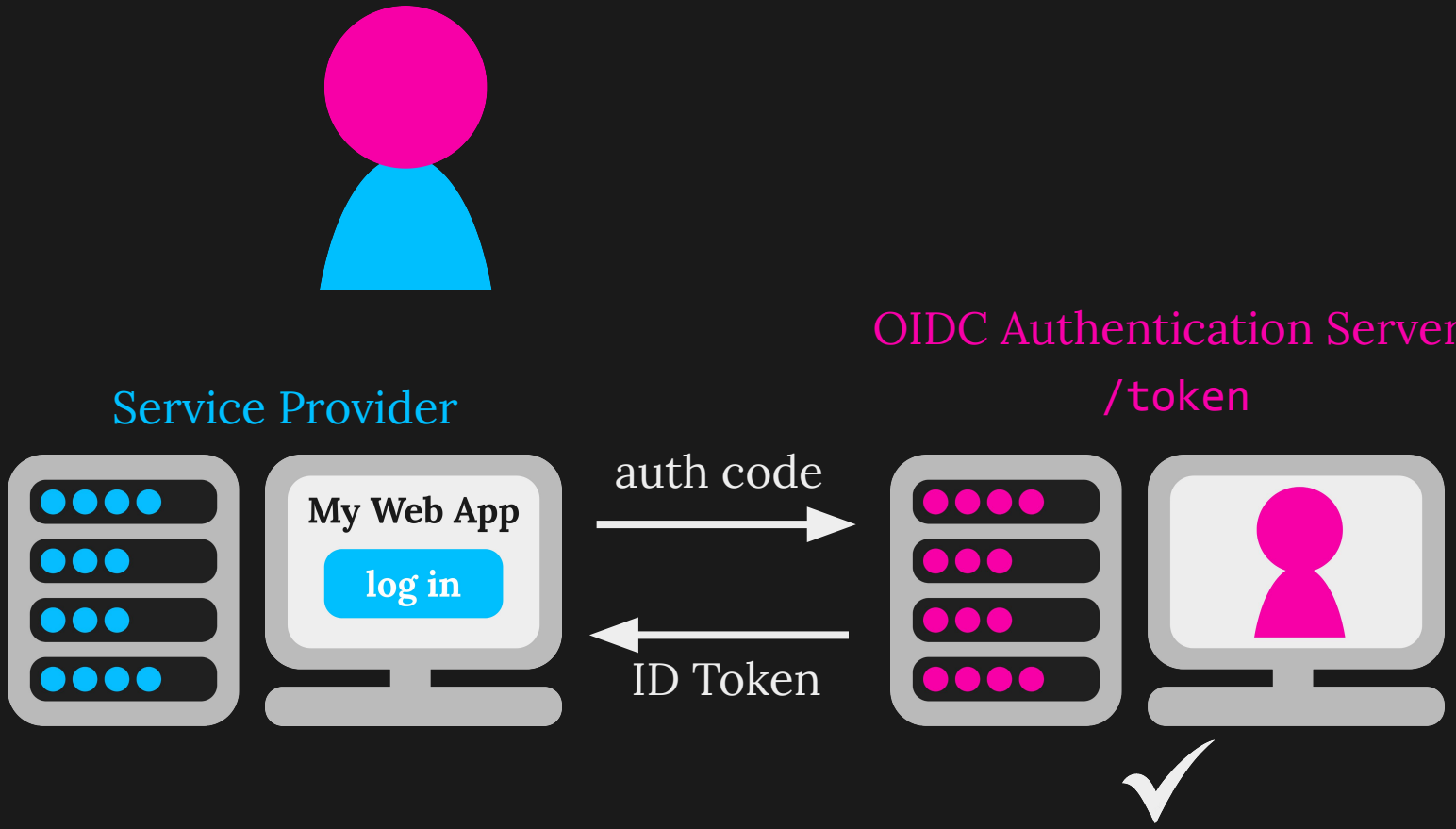
@eli@hachyderm.io



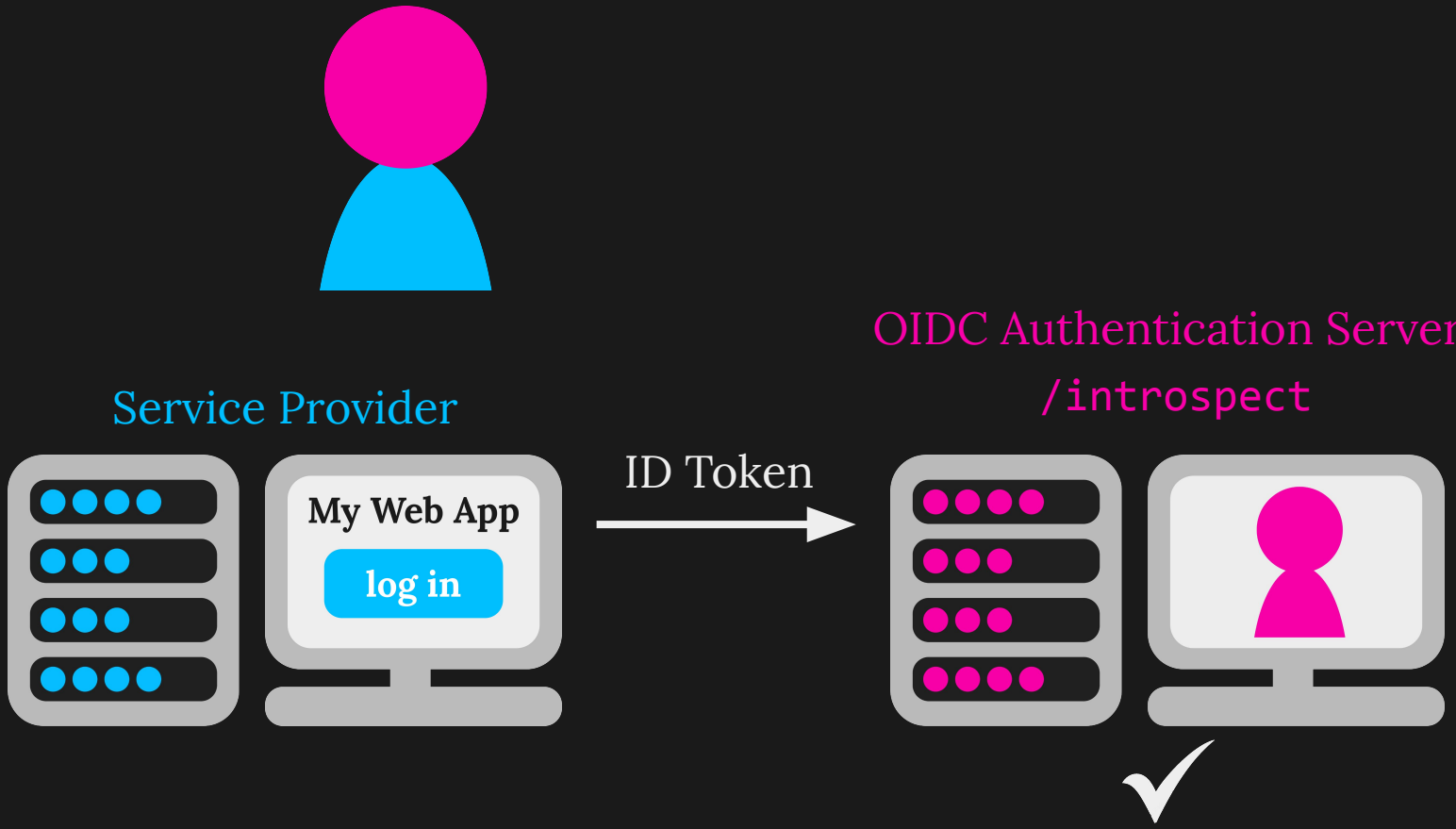
@eli@hachyderm.io



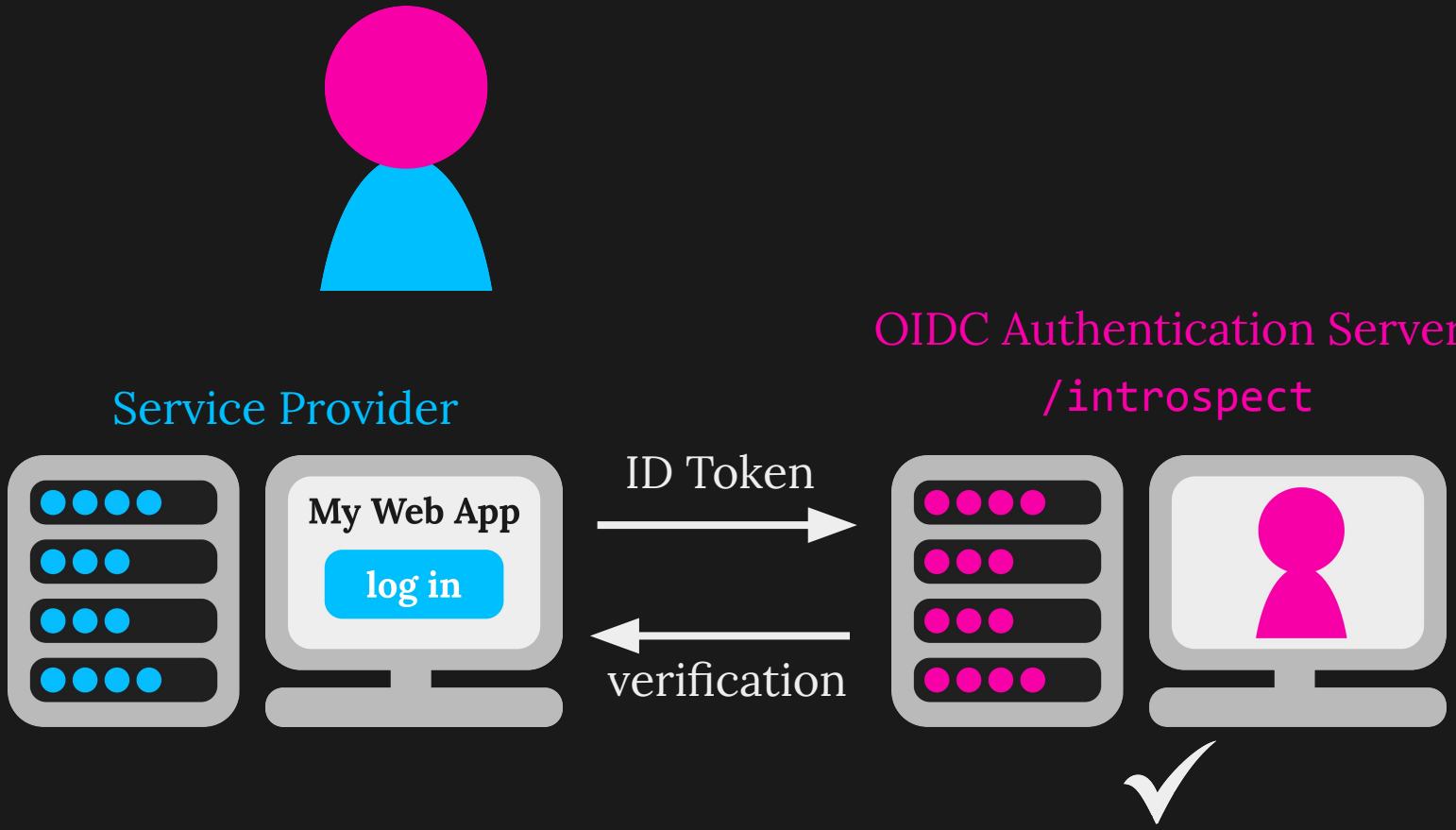
@eli@hachyderm.io



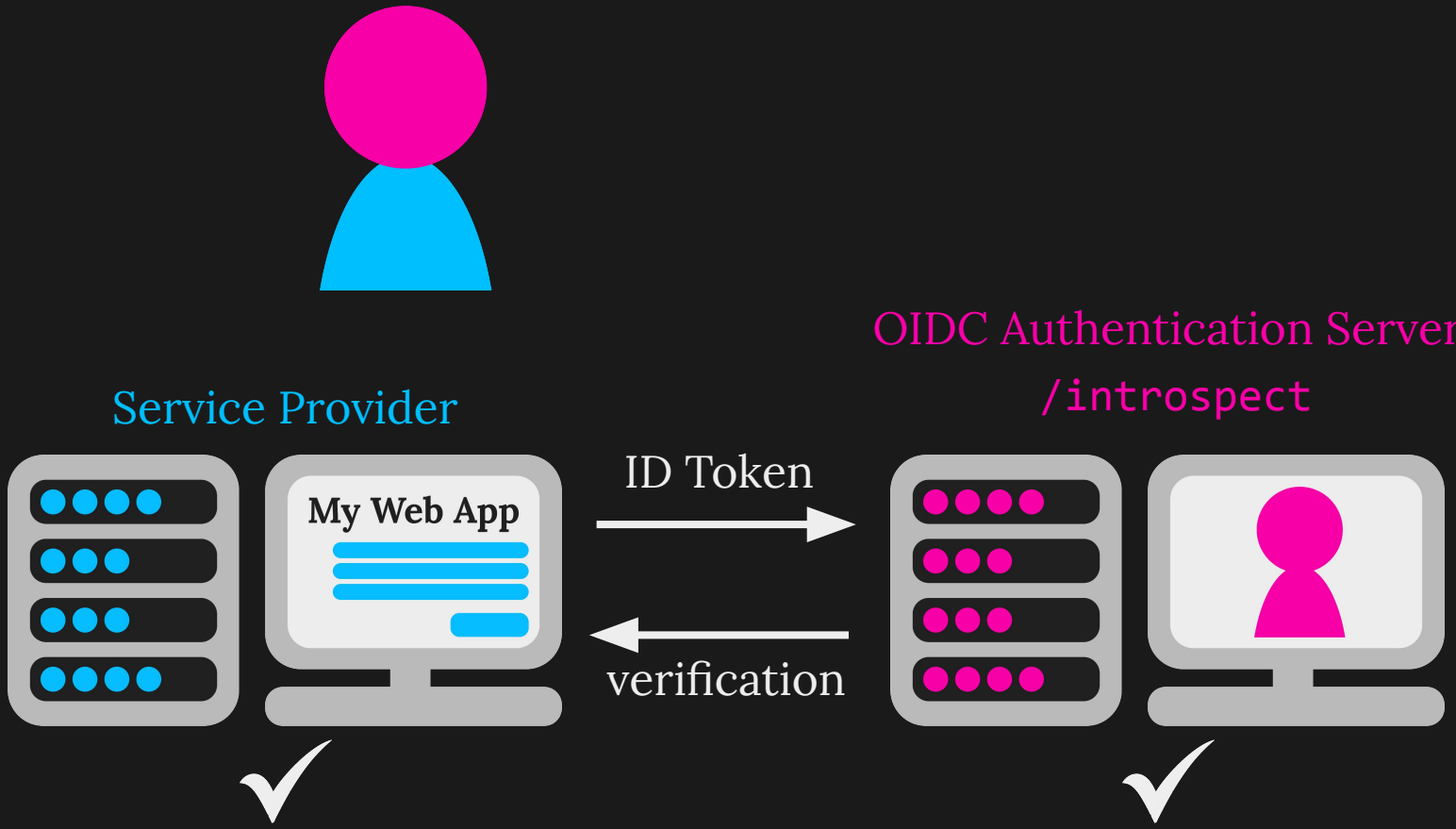
@eli@hachyderm.io



@eli@hachyderm.io



@eli@hachyderm.io



@eli@hachyderm.io

Service Provider & Identity Provider communicate directly

Service Provider & Identity Provider communicate directly

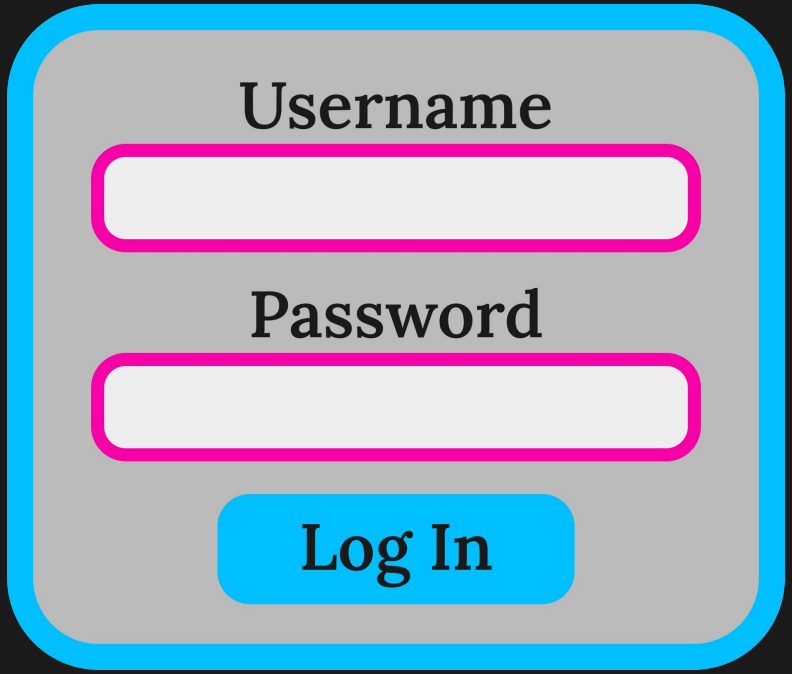
extensible with other endpoints such as `/userinfo`

What makes an authentication system good?

@eli@hachyderm.io

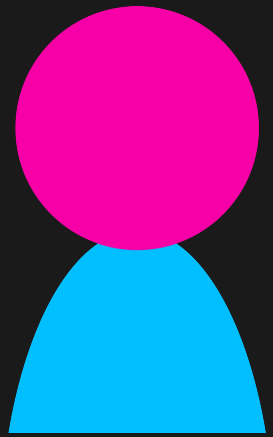
Security

@eli@hachyderm.io



@eli@hachyderm.io

Ip9q7nv3iueyt9

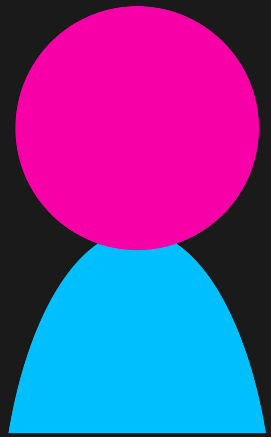


@eli@hachyderm.io

Ip9q7nv3iueyt9

ijgnbAf894bCC

0h935uMy587



pq9nV3y45jhy

jhLIUG32nv4

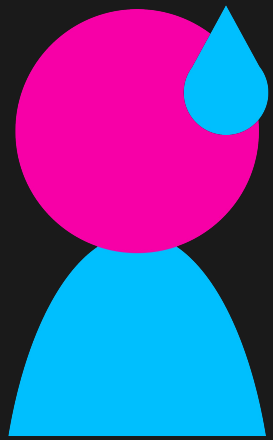
84ynvLd7hg

@eli@hachyderm.io

Ip9q7nv3iueyt9

ijgnbAf894bCC

0h935uMy587



pq9nV3y45jhy

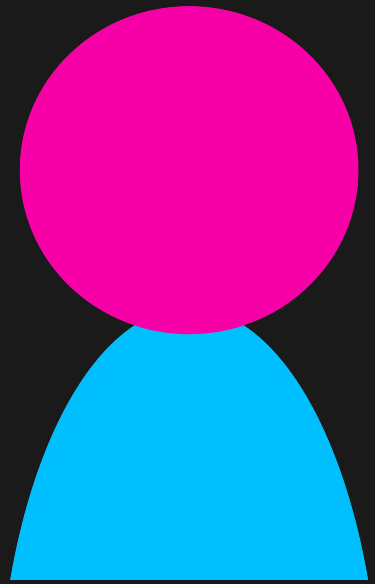
jhLIUG32nv4

84ynvLd7hg

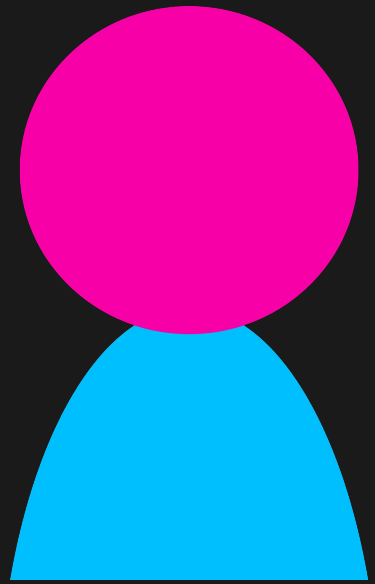
@eli@hachyderm.io

If something is hard to use securely, it's insecure

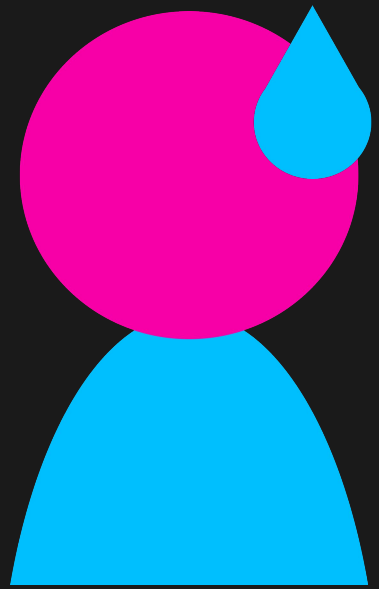
@eli@hachyderm.io



@eli@hachyderm.io



@eli@hachyderm.io



@eli@hachyderm.io

Security

Ease of use

@eli@hachyderm.io

<saml:Issuer> ... </saml:Issuer>

< saml:Issuer > ... < / saml:Issuer >

@eli@hachyderm.io

If something is hard to implement securely, it's insecure

@eli@hachyderm.io

Security

Ease of use

Ease of implementation

@eli@hachyderm.io

What have we learned?

@eli@hachyderm.io

What authentication is

What authentication is

... and why we need it

@eli@hachyderm.io

What authentication is

... and why we need it

How we do it

@eli@hachyderm.io

What authentication is

... and why we need it

How we do it

... and why it's hard

@eli@hachyderm.io

What authentication is

... and why we need it

How we do it

... and why it's hard

How to do it well

@eli@hachyderm.io

What authentication is

... and why we need it

How we do it

... and why it's hard

How to do it well

... and that usability matters

@eli@hachyderm.io

You Shall Not Password

Modern Authentication for Web Services

Eli Holderness
they / them / theirs

@eli@hachyderm.io