



From our self-build authentication to Ory

—

David ALEXANDRE
Co-founder and CTO at Wildcard
david.alexandre@w6d.io



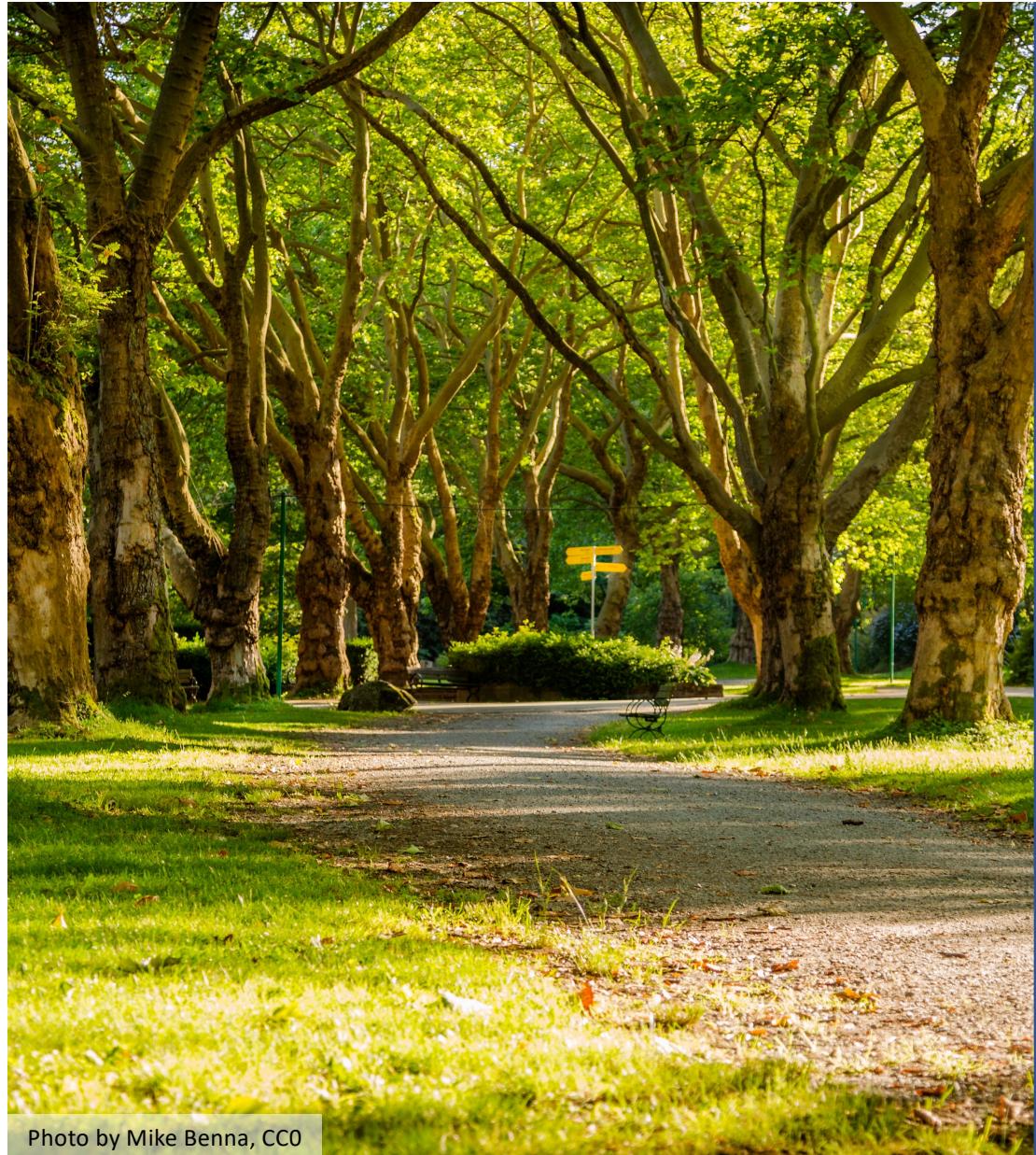


Photo by Mike Benna, CC0

Our use case

About us

Why Ory

The future user experience

How we do it

Q&A

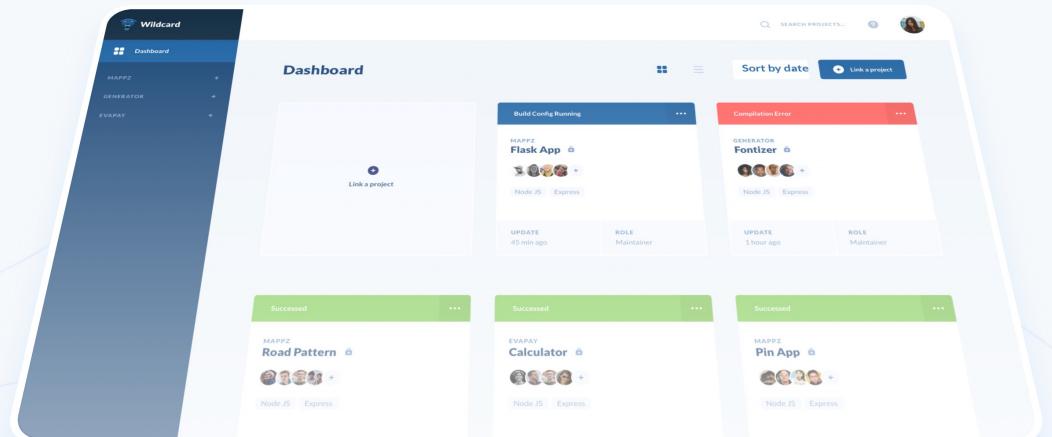




Build, test & deploy app with NoCode

The hassle-free NoCode Platform
to deliver your cloud native apps

TRY FOR FREE NOW



The screenshot shows the Wildcard CI/CD pipeline interface. On the left is a dark sidebar with project names: W6D and W6D/FRONTEND. The main area displays five completed pipelines (Pipeline 10, Pipeline 9, Pipeline 8, Pipeline 7, Pipeline 6) and one failed pipeline (Pipeline 9). Each pipeline card includes details like commit message, branch, date, duration, and a success/failure status.

Pipeline 10
7828635b

COMMIT MESSAGE	BRANCH	DATE	DURATION
update readme.md	develop	Oct 20, at 11:56am	3min 11s

Pipeline 9
76415e59

COMMIT MESSAGE	BRANCH	DATE	DURATION
update readme.md	develop	Oct 20, at 11:52am	3min 59s

Pipeline 8
6fa5dac4

COMMIT MESSAGE	BRANCH	DATE	DURATION
feat: add triggers call	develop	Oct 19, at 2:19pm	5min 9s

Pipeline 7
7c6b6074

COMMIT MESSAGE	BRANCH	DATE	DURATION
perf: shrink the file number in pipeline package	develop	Oct 19, at 11:42am	3min 53s

Pipeline 6
54e1587c

COMMIT MESSAGE	BRANCH	DATE	DURATION
update readme.md	develop	Oct 18, at 10:34am	3min 10s

Pipeline 9
Failed

COMMIT MESSAGE	BRANCH	DATE	DURATION
update readme.md	develop	Oct 20, at 11:52am	3min 59s



PIPLINES PROJECT BUILD CI/CD

Search a project...

W6D **cicd** PROD

Your app is not deployed yet.

CHOOSE YOUR BRANCH AND CREATE YOUR PIPELINE

develop

Enter a branch name

Unit test

Git leaks

codecov / sonarqube

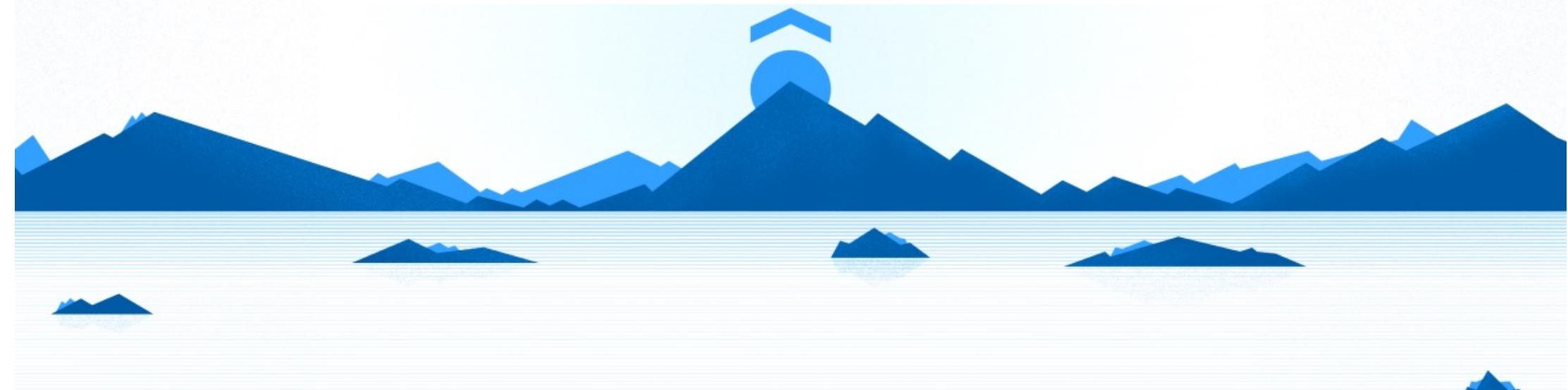
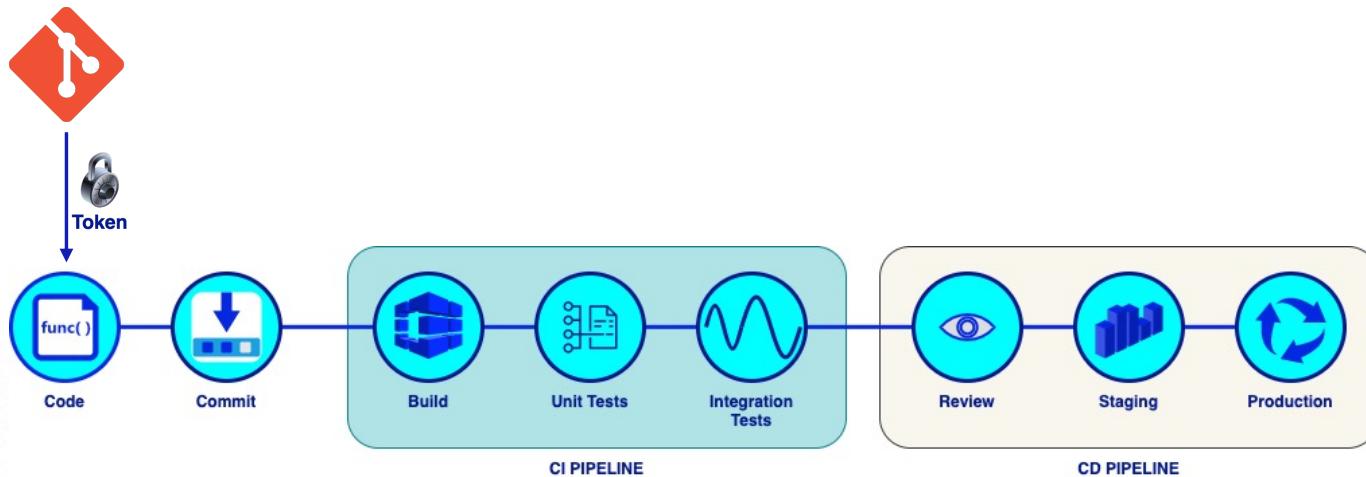
Build docker

Scan image vulnerabilities

Scan web app vulnerabilities (owasp compliance)

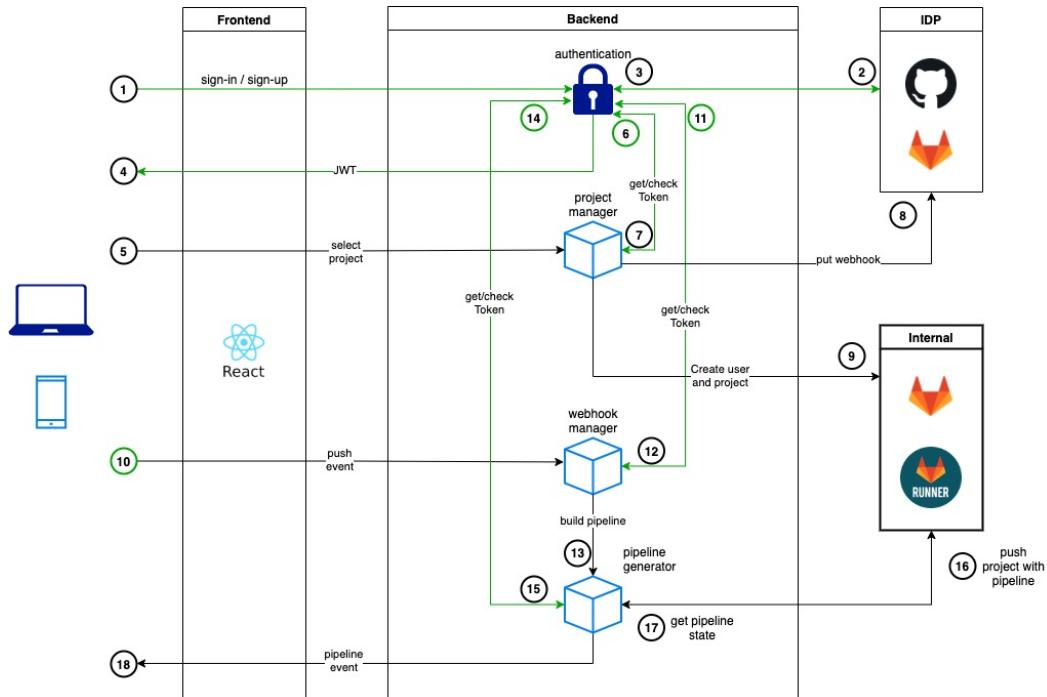
Scan CVE vulnerabilities

Deploy

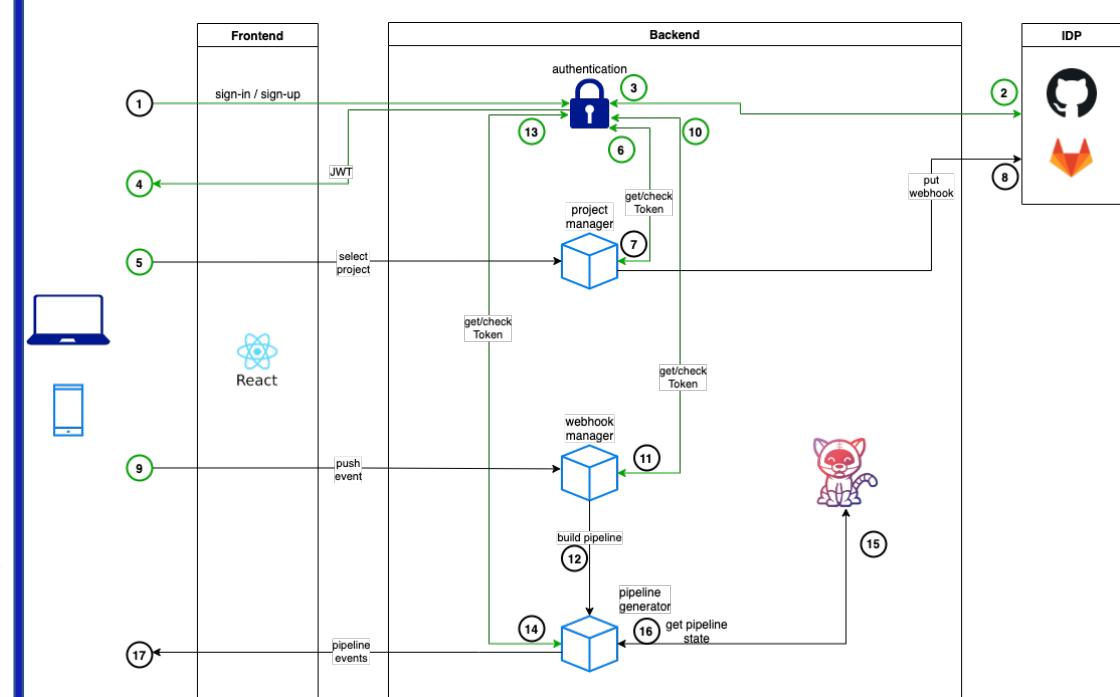


Our history

BEFORE



NOW



Why Ory ?

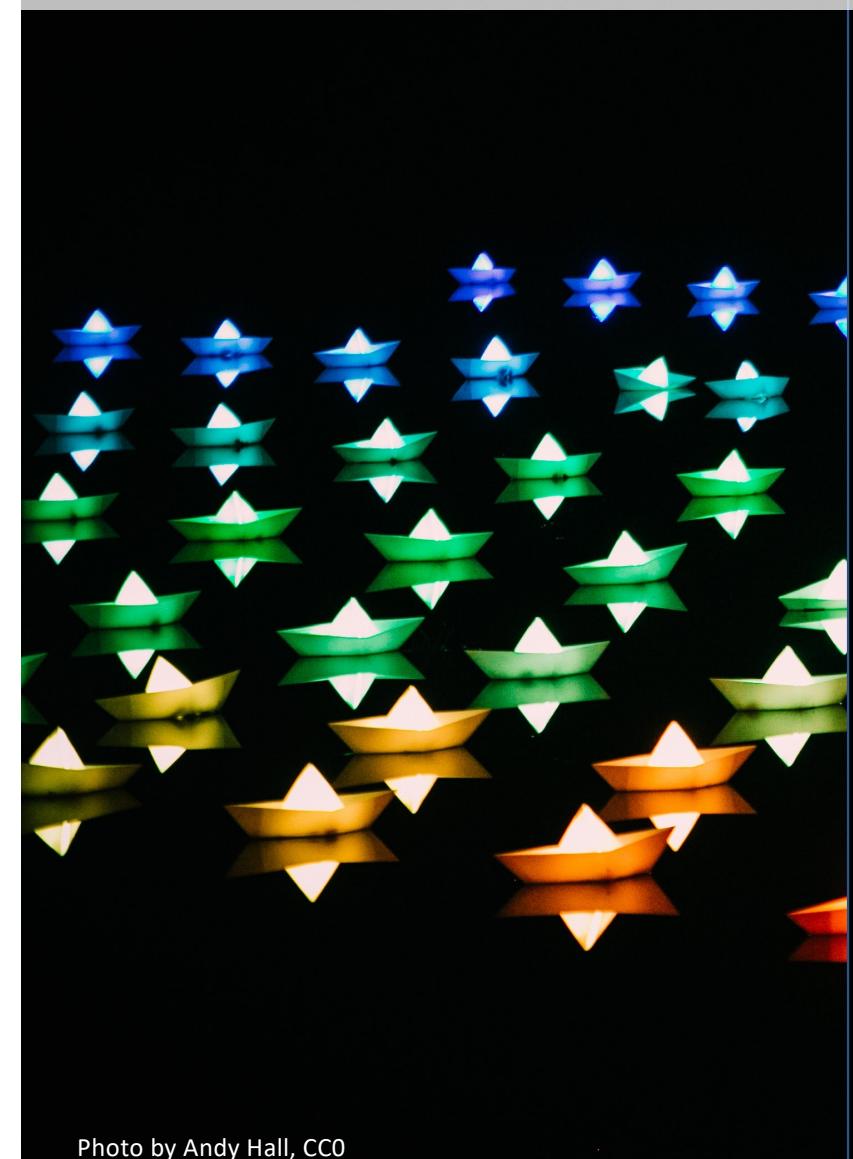
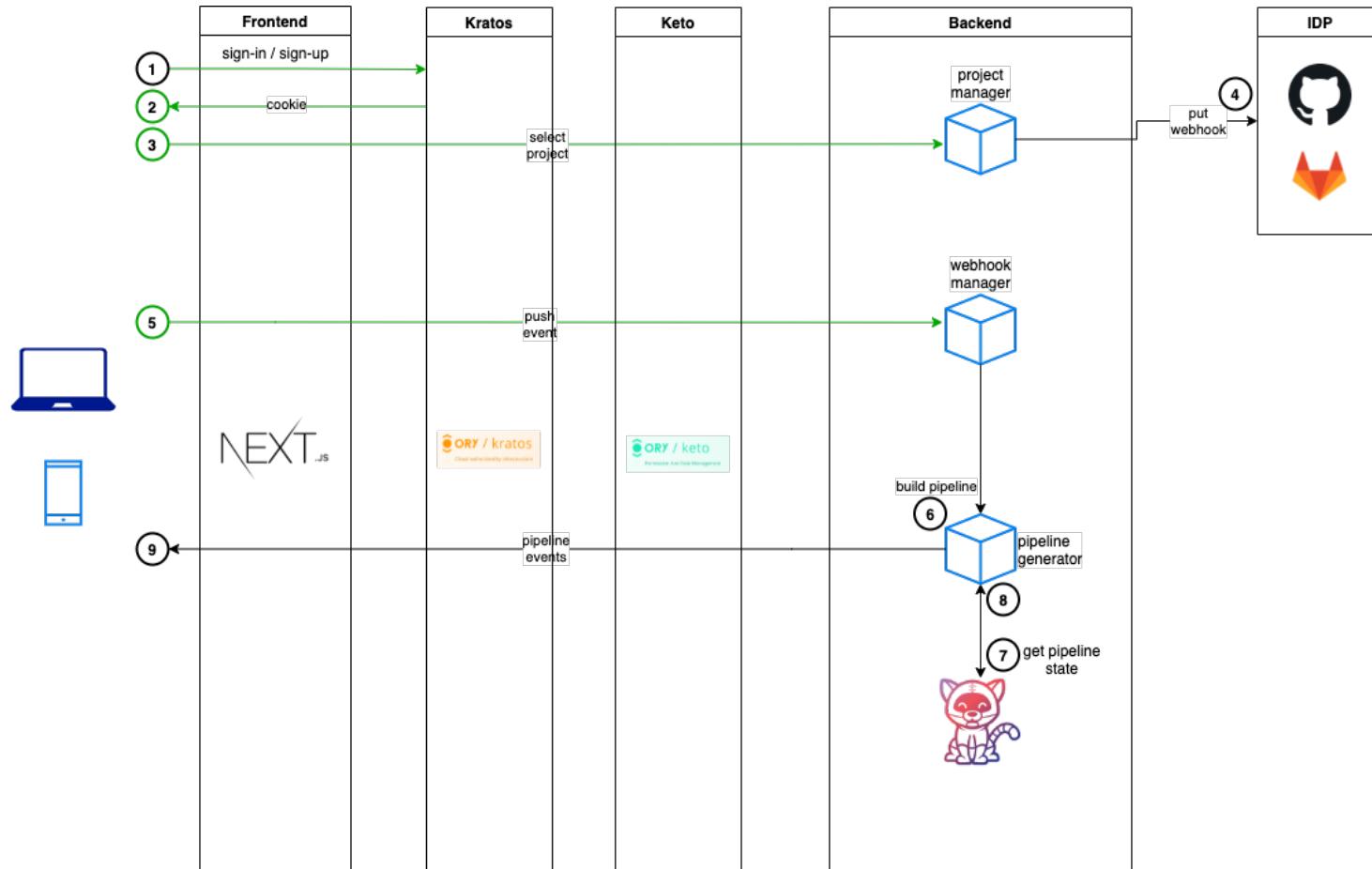


Photo by [Andy Hall](#), CCO



The future user experience



The future user experience



Photo by [Mikhail Vasilyev](#), CC0

Provider Management

FIRSTNAME LASTNAME

EMAIL ADDRESS

CHOOSE A PASSWORD

CONFIRM PASSWORD

 Github
Synced with account@email.com Unlink

 Gitlab
Gitlab.com Link

 Bitbucket
Bitbucket.com Link

Save

The future user experience

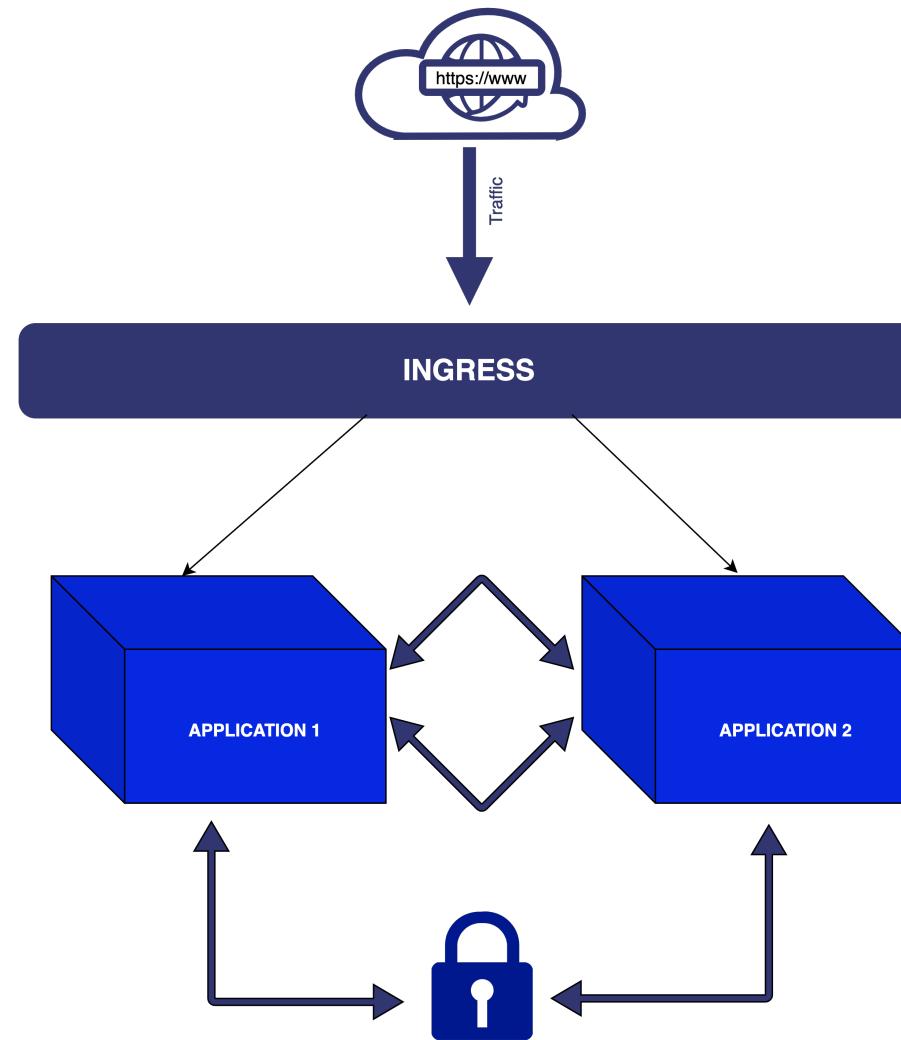
- Access management
- Team management

A screenshot of a modal dialog titled "Add member". It contains fields for FIRSTNAME (Patrick), LASTNAME (Sanchez), and EMAIL ADDRESS (b.sanchez). Below these, a dropdown menu shows "Viewer" selected under "ROLE IN THE ORGANISATION". A checkbox labeled "Apply by default this role to all existing scopes in the organization." is checked. Below this, four scope entries are listed: ScopeName 1 (Viewer), ScopeName 2 (Viewer), ScopeName 3 (No Access), and ScopeName 4 (Admin). At the bottom are "Cancel" and "Create" buttons.

A screenshot of the "Members" page. The top navigation bar includes "OrgaName", "Summary", "Projects", "Databases", "Members", and "Finances". Below this, a header row shows "All members", "Owner and Admins (2)", "Contributor (1)", "Viewer (1)", and "Finance (1)". There are "Filter by scope" and "Add Member" buttons. The main table lists six users:

	PERSON	MAIN ROLE	ROLE SCOPES	
<input type="checkbox"/>	Daniel Padilla daniel.padilla@hotmail.com	OWNER		...
<input type="checkbox"/>	Myrtle Dennis myrtle.dennis@gmail.com	CONTRIBUTOR	ADMIN ON + SCOPE1 FE ON + SCOPE2	...
<input type="checkbox"/>	Evan McCormick evan.mcc@yahoo.com	ADMIN		...
<input type="checkbox"/>	Bernard Ford b.ford2@dare.net	VIEWER		...
<input type="checkbox"/>	Christina Jacobs christina.jacobs@gmail.com	FINANCER		...

How we do it



How we do it – Step by step

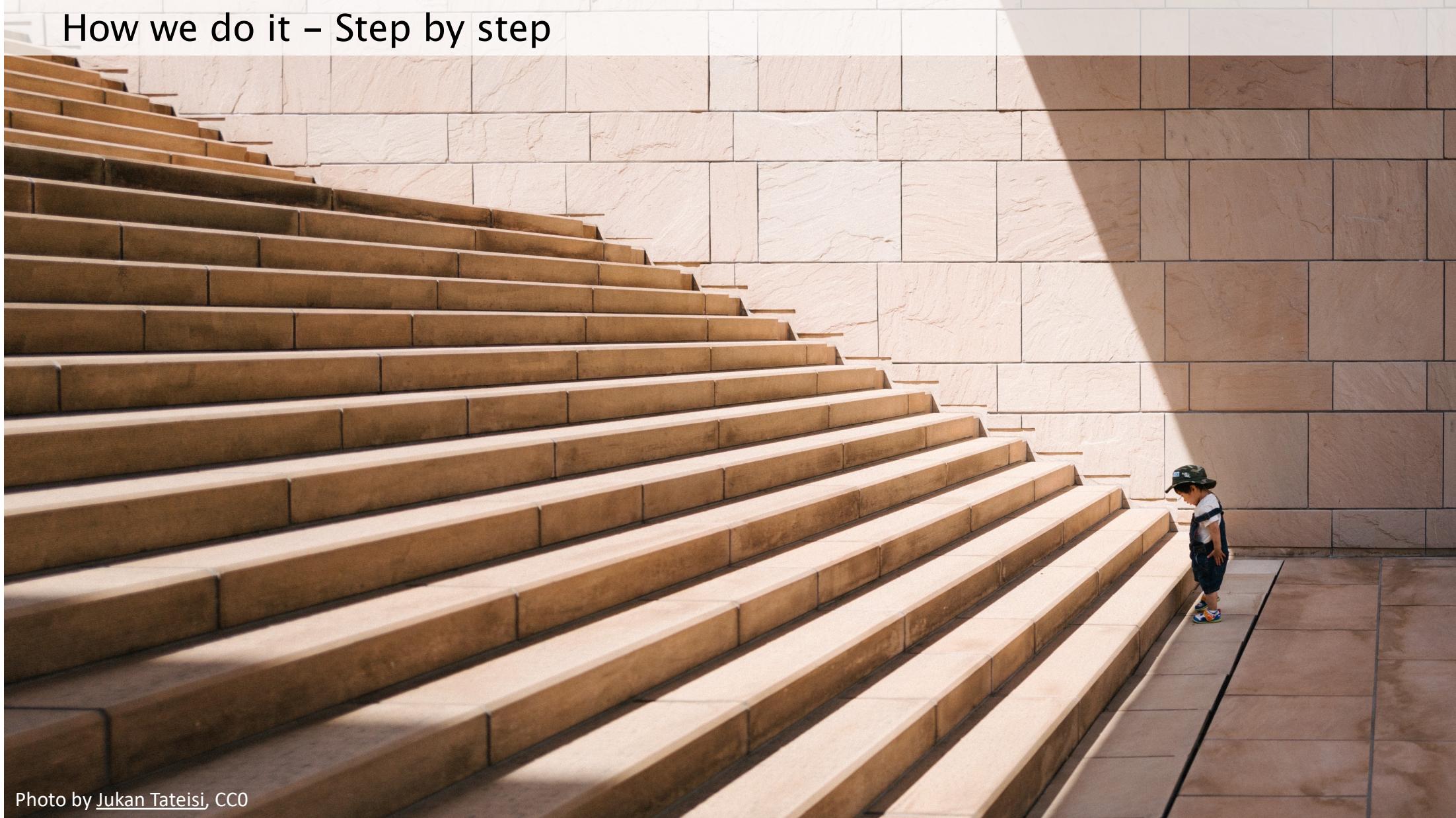
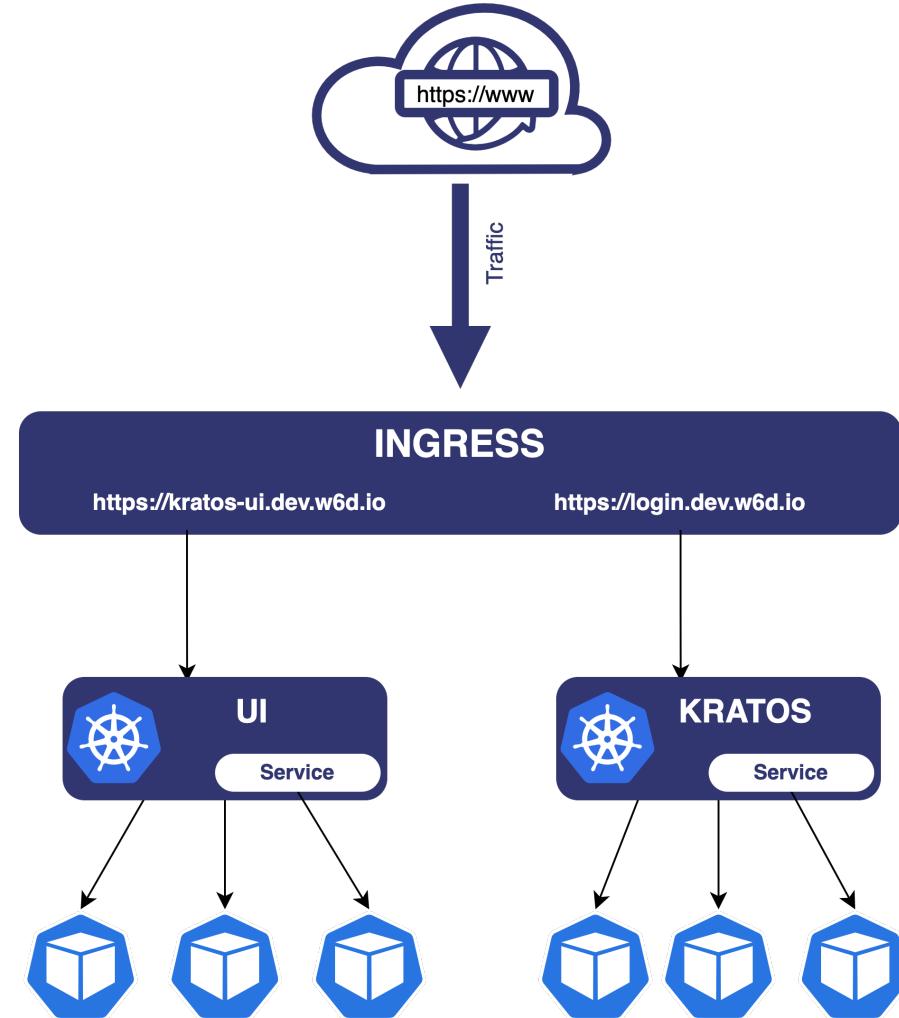


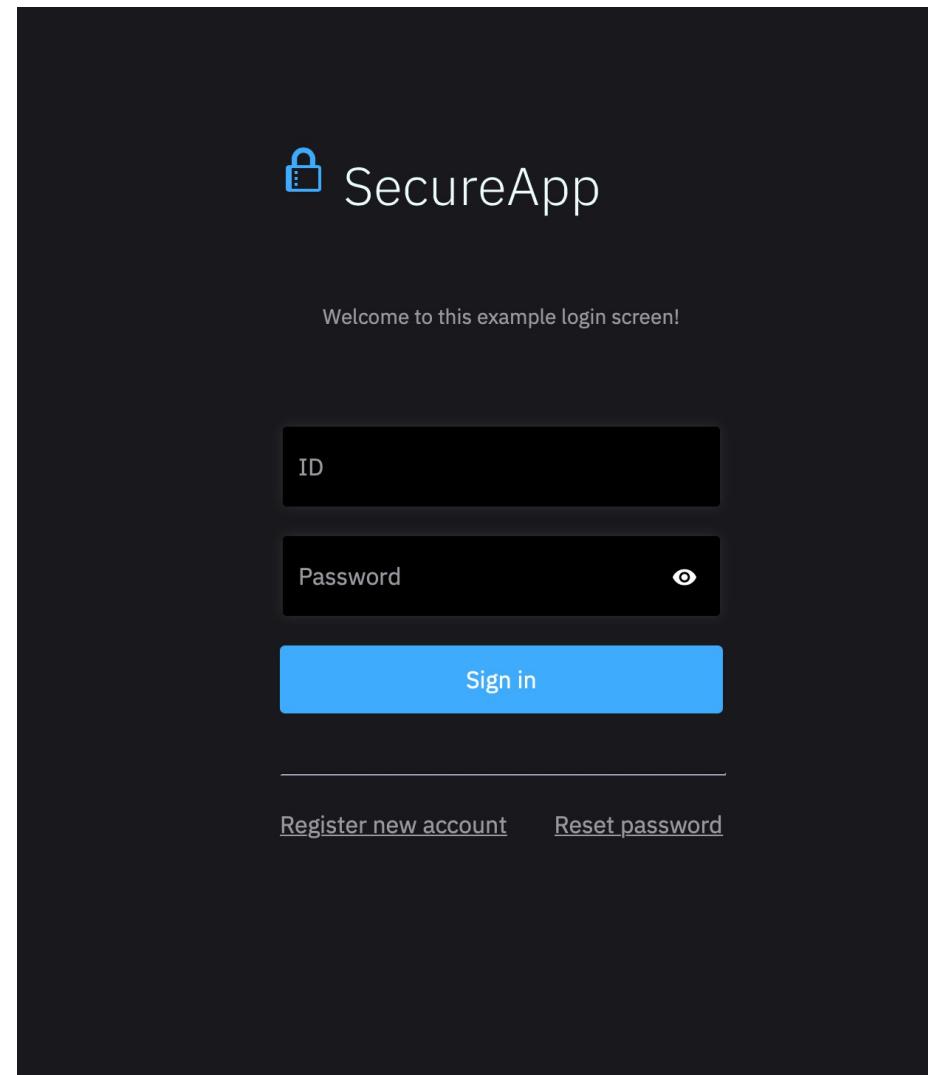
Photo by [Jukan Tateisi](#), CC0

STEP 1

Rolling out QuickStart



STEP 2 Tests



STEP 3

OIDC

Configuration

```
30     whitelisted_return_urls:
31         - http://127.0.0.1:8080
32         - http://127.0.0.1:4433
33         - http://localhost:8080
34         - "*"
35
36     methods:
37         password:
38             enabled: true
39         oidc:
40             enabled: true
41             config:
42                 providers:
43                     gitlab:
44                         provider: gitlab
45                         client_id: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxx'
46                         client_secret: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxx'
47                         mapper_url: file:///kratos/mapper/gitlab-w6d.data-mapper.jsonnet
48                         scope:
49                             - api
50                             - read_user
51                             - read_repository
52                             - openid
53                             - profile
54                             - email
55                     github:
56                         provider: github
57                         client_id: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxx'
58                         client_secret: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxx'
59                         mapper_url: file:///etc/config/kratos/github.data-mapper.jsonnet
60                         scope:
61                             - user:email
62                             - repo
63                         flows:
64                             error:
65                                 ui_url: http://127.0.0.1:8080/error
66
67             settings:
68                 ui_url: http://127.0.0.1:8080/settings
69                 privileged_session_max_age: 15m
```

STEP 4

Tests

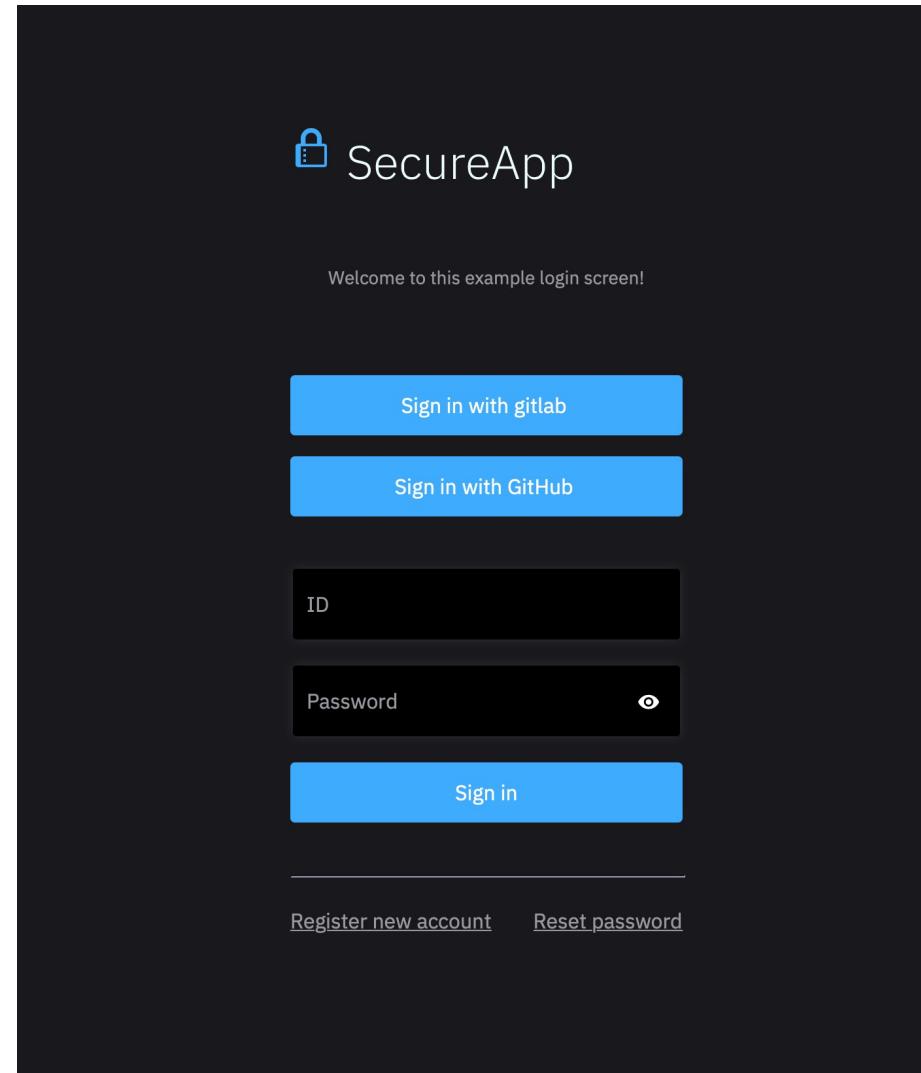
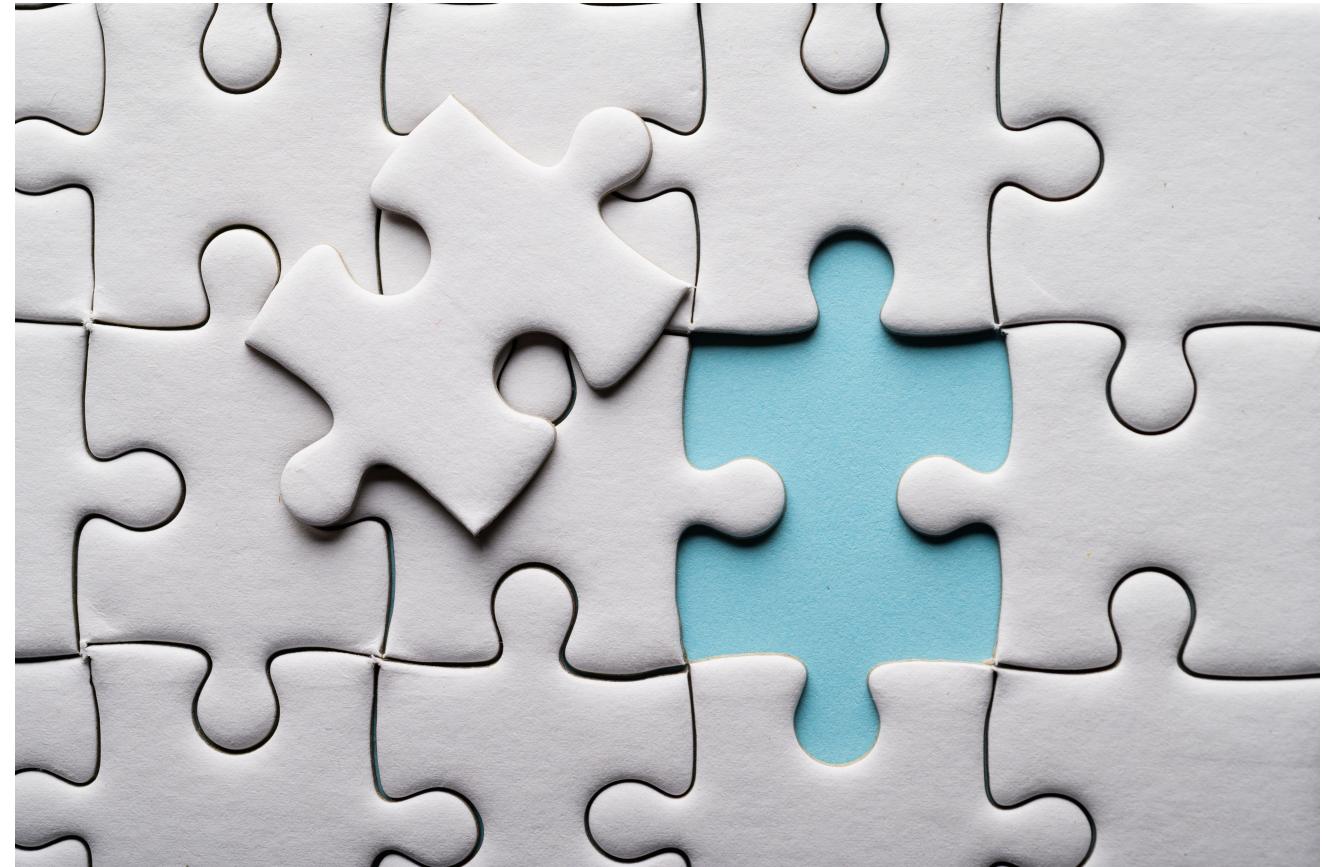




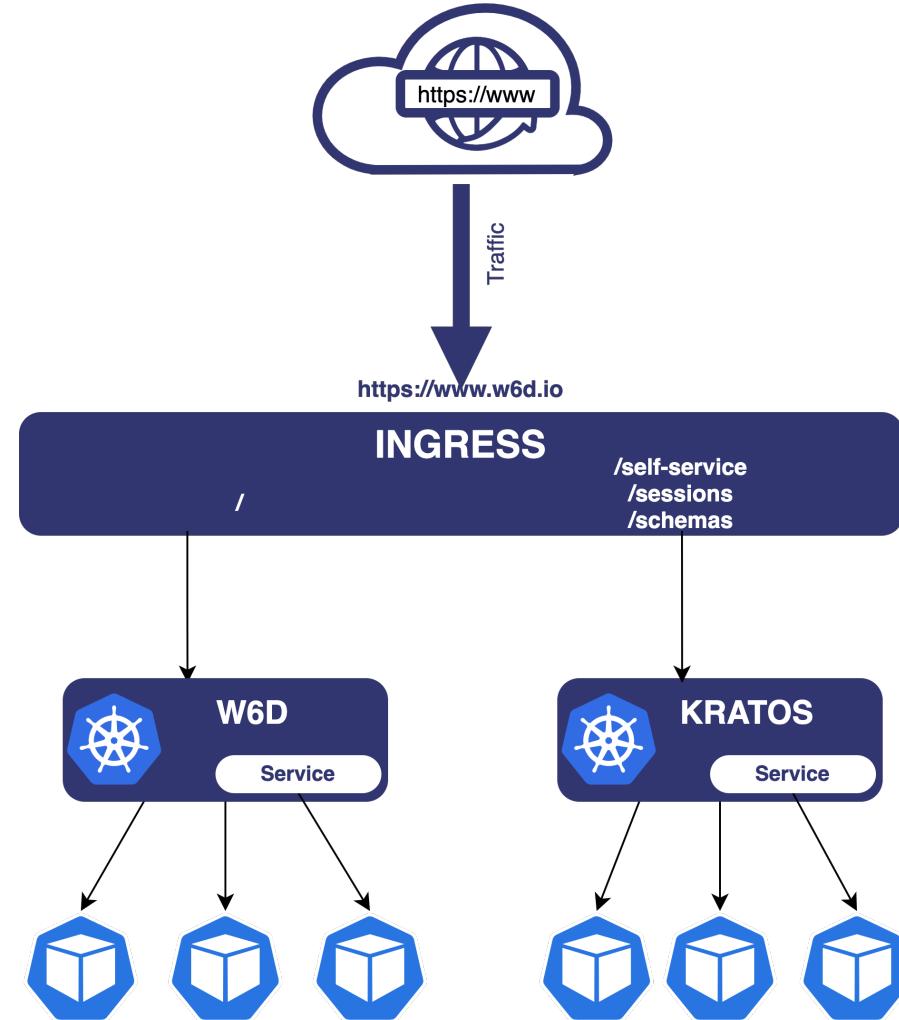
Photo by [Aaron Burden](#), CC0

Something missing !

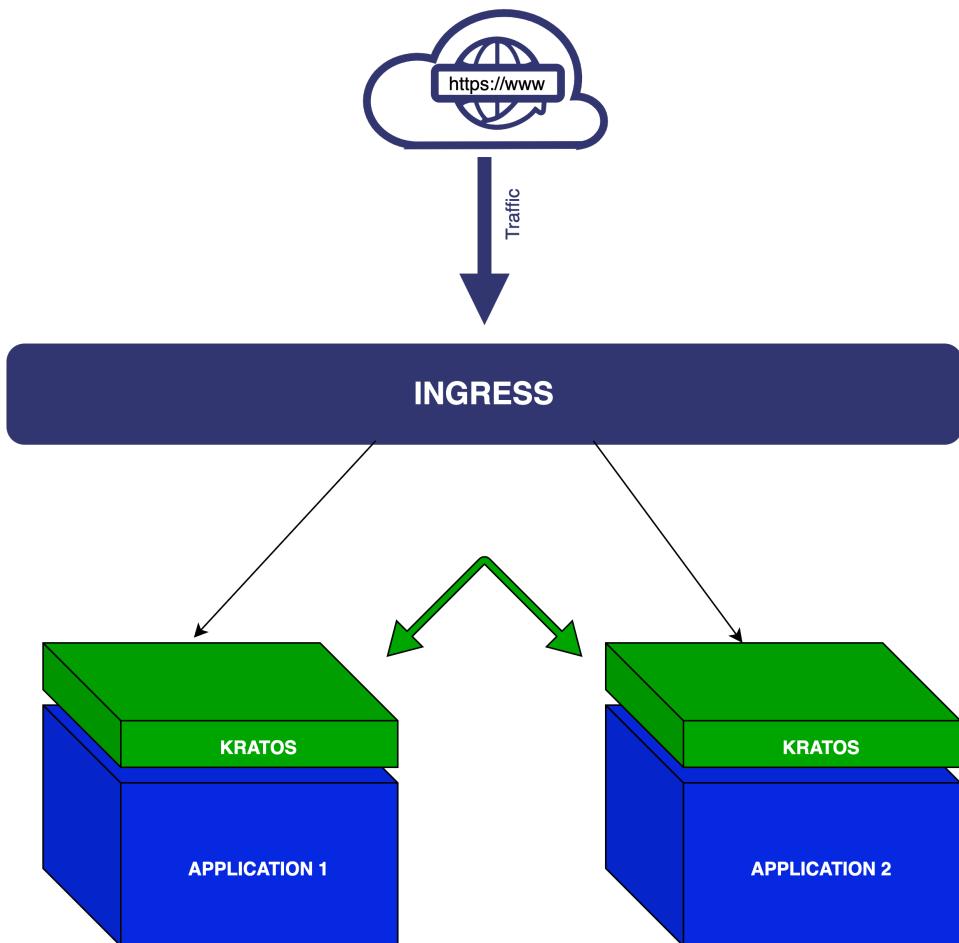
Our
contribution



Last Kratos step implementation

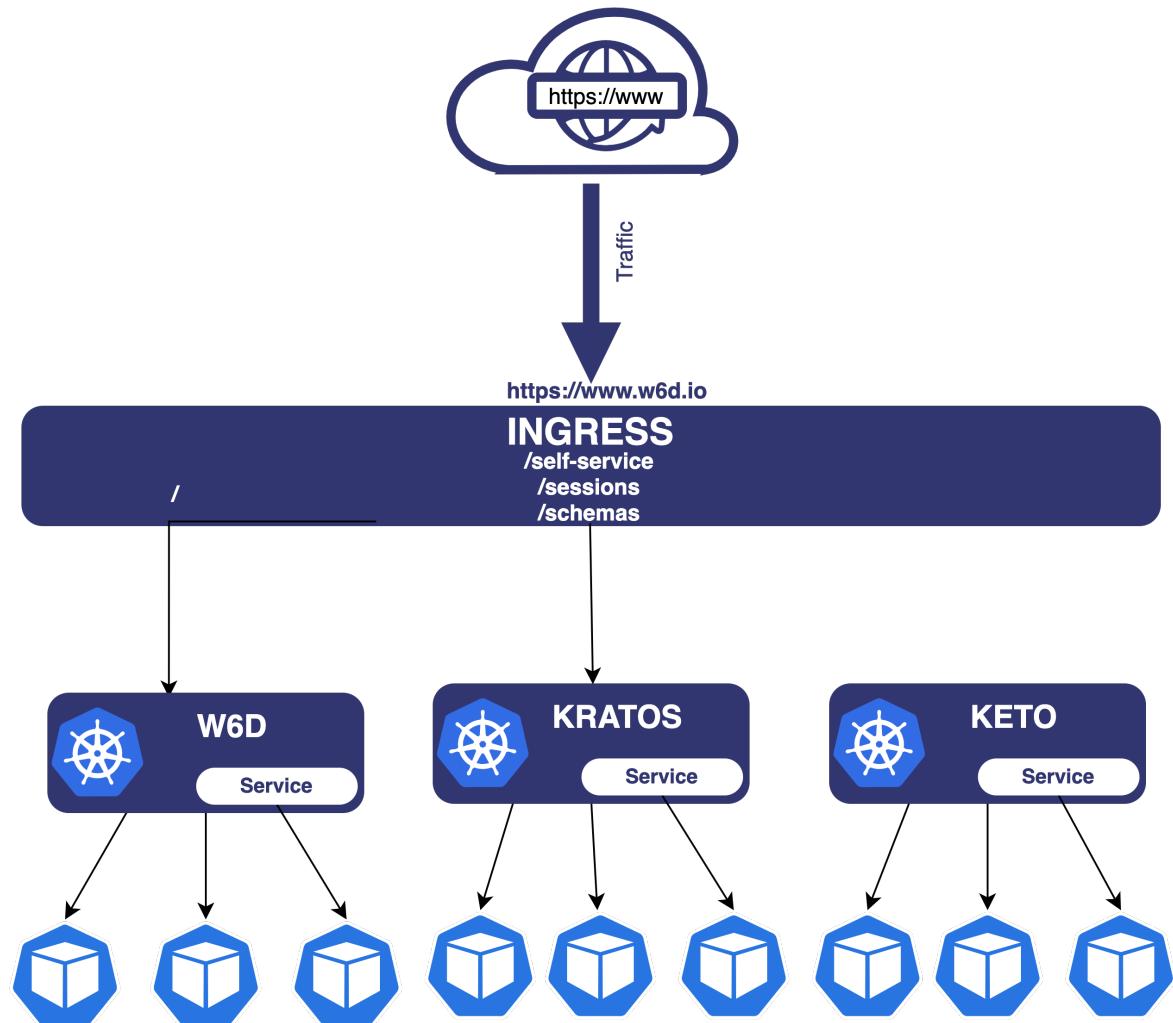


Now



```
1 package kratos
2
3 import (
4     "context"
5     "net/http"
6     "net/url"
7
8     "github.com/w6d-io/x/errorx"
9
10    client "github.com/ory/kratos-client-go"
11    "github.com/ory/kratos/identity"
12 )
13
14 type Conn struct {
15     // Address is the the address to the kratos micro service
16     Address string `json:"address" mapstructure:"address"`
17 }
18
19 type Helper interface {
20
21     // GetSessionFromHTTP is used to check if the session cookie is active ( ex: session.GetActive() )
22     // and also return user information
23     // If session is not set, return a nil session with StatusBadRequest and error
24     // If kratos is unreachable or an other issues, return nil session with statusCode of the call and error--go
25     GetSessionFromHTTP(ctx context.Context, req *http.Request) (*client.Session, error)
26
27     // GetSessionFromGRPCCtx is used to forward a session stock into a context.
28     // It checks if session on context is present
29     // If session is not set, return a nil session with StatusBadRequest and error
30     // If kratos is unreachable or an other issues, return nil session with statusCode of the call and error--go
31     GetSessionFromGRPCCtx(ctx context.Context) (*client.Session, error)
32
33     // GetIdentityFromHTTP is used to get the identity who correspond to the user id on kratos service
34     // If kratos is unreachable or an other issues, return nil session with statusCode of the call and error--go
35     GetIdentityFromHTTP(ctx context.Context, id string) (*identity.Identity, error)
36
37     // GetIdentityFromAPI is used to get the identity who correspond to the user id on kratos service
38     // If kratos is unreachable or an other issues, return nil session with statusCode of the call and error--go
39     GetIdentityFromAPI(ctx context.Context, id string) (*client.Identity, error)
40
41     // GetIdentityFromCtxHTTP gets the session from context and retrieve the identity ID
42     // to make the http call
43     GetIdentityFromCtxHTTP(ctx context.Context) (*identity.Identity, error)
44
45     // GetIdentityFromCtxApi gets the session from context and retrieve the identity ID
46     // to make the api call
47     GetIdentityFromCtxApi(ctx context.Context) (*client.Identity, error)
48
49     // GetTokenByHttp returns all tokens linked with the provider
50     GetTokenByHttp(ctx context.Context, provider string) (*Provider, error)
51
52     // GetTokensByHttp returns all tokens linked with the provider
53     GetTokensByHttp(ctx context.Context) ([]Provider, error)
54 }
55
56 type Provider struct {
57     TokenID      string `json:"initial_id_token"`
58     Subject      string `json:"subject"`
59     Provider     string `json:"provider"`
60     AccessToken  string `json:"initial_access_token"`
61     RefreshToken string `json:"initial_refresh_token"`
62 }
```

Our next move



Our next move

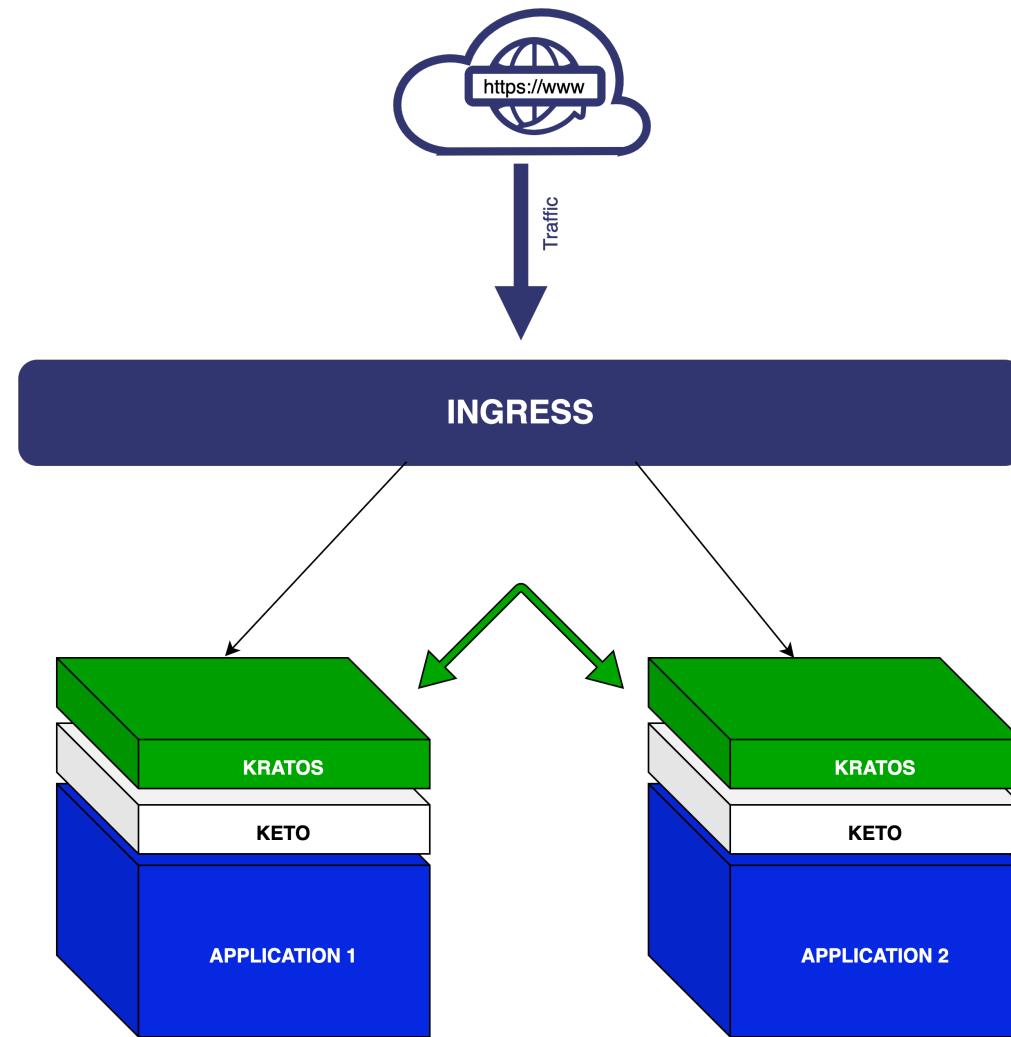




Photo by [Alvan Nee](#), CC0

Thank you

Questions ?