Making Security Usable:

Tales of Product Engineering
...in a Security Company

@vixentael

#data_security

#cryptography

#product_thinking

#product_design

@vixentael

Feel free to reach me with security questions.

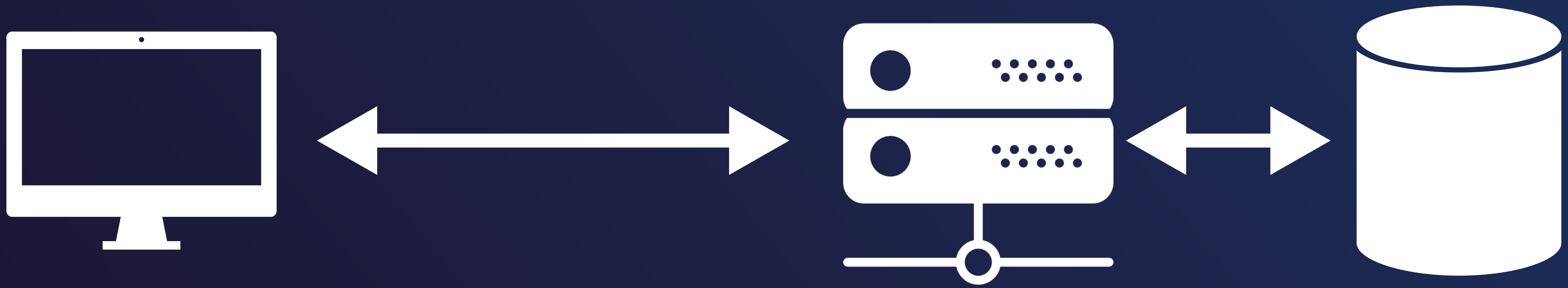
I do check my inbox :)

Product Engineer

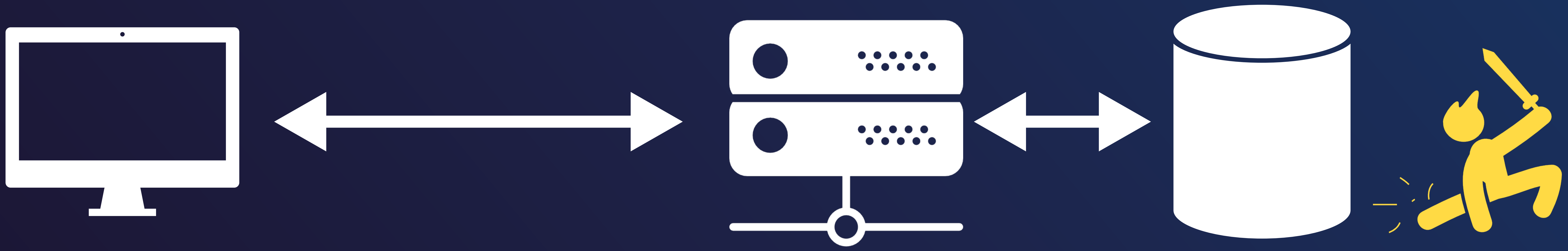
COSSACK LABS

I. The story

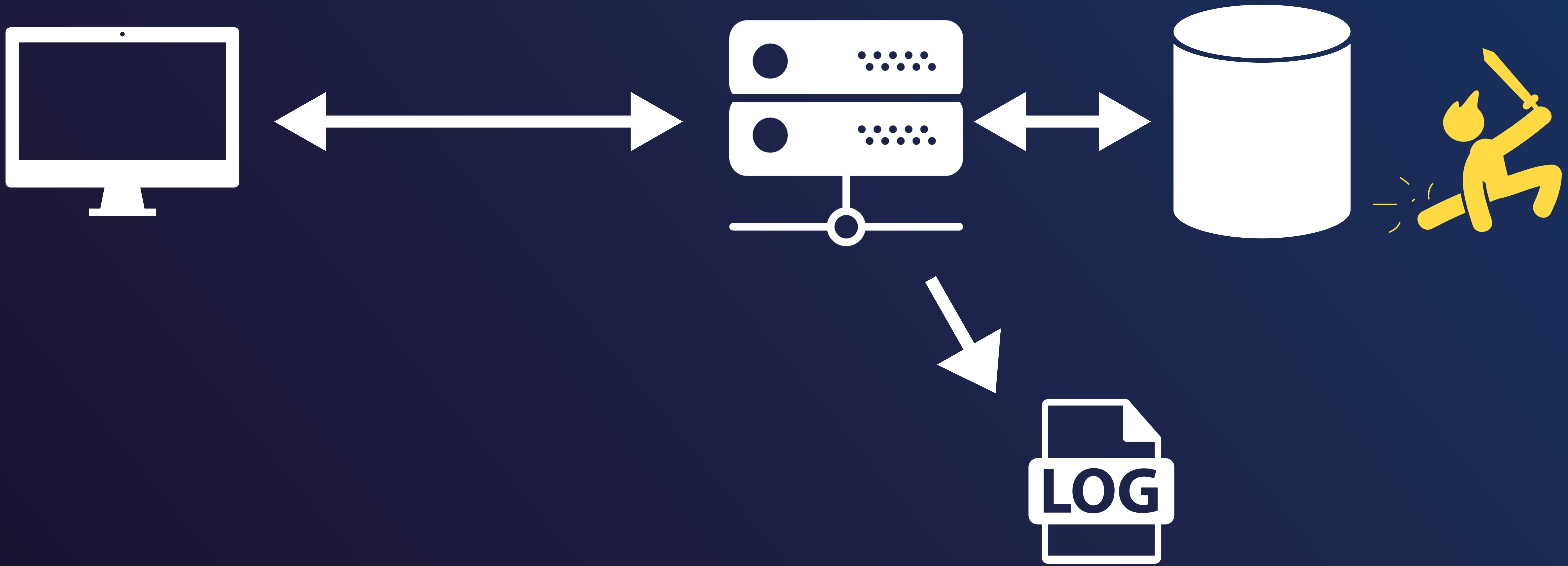
A long time ago in a galaxy far, far away…



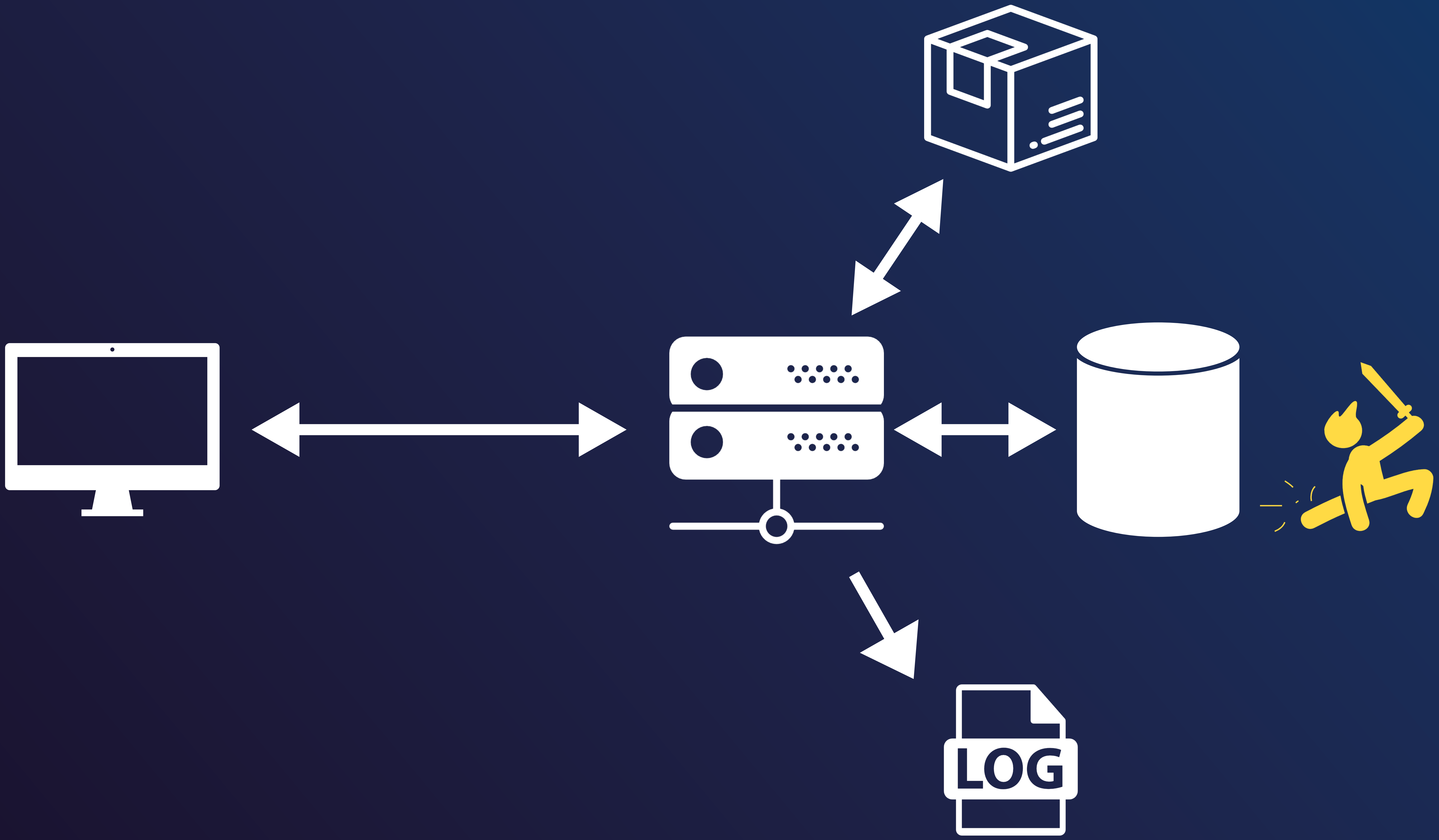
@vixentael



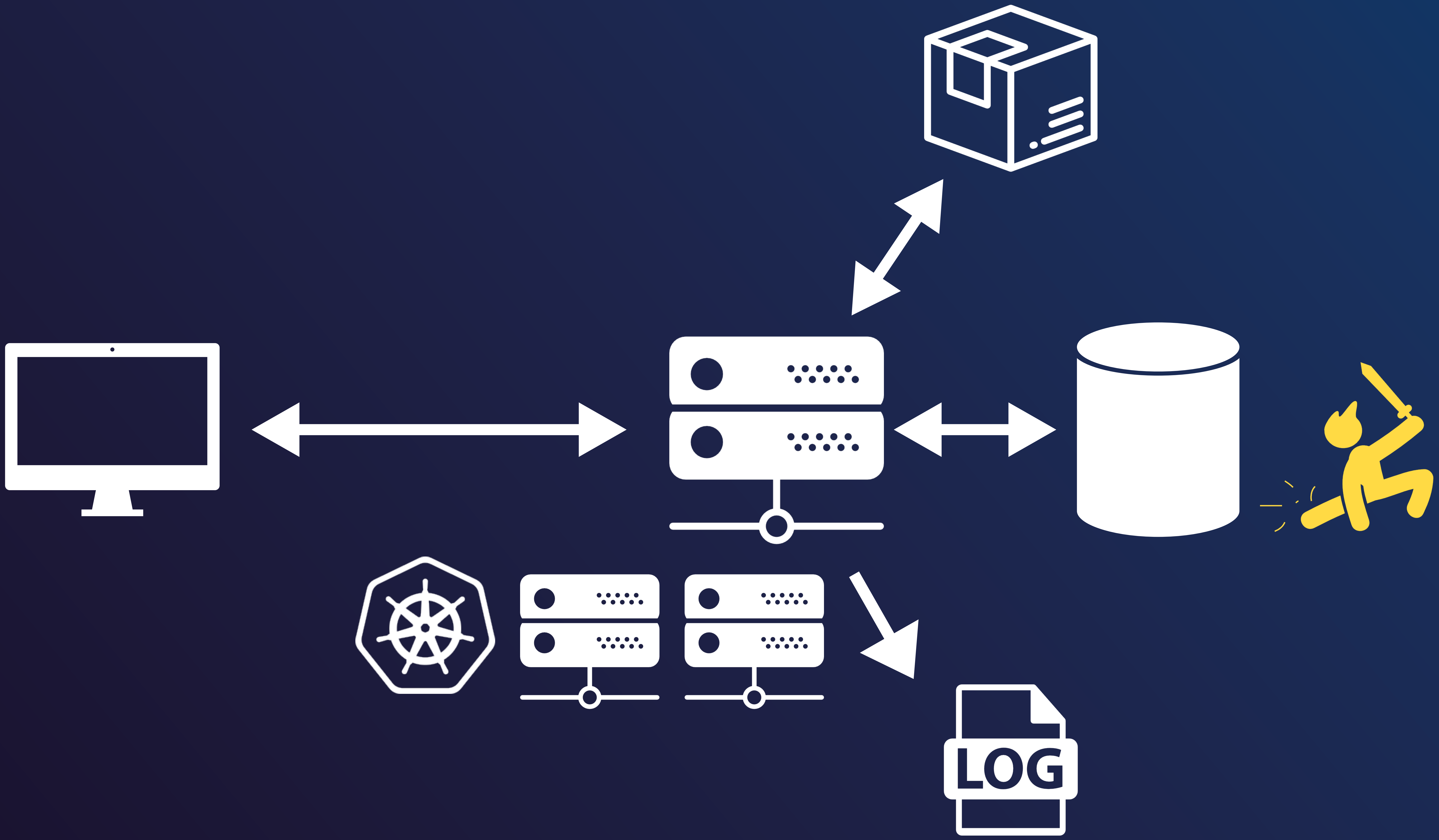
@vixentael



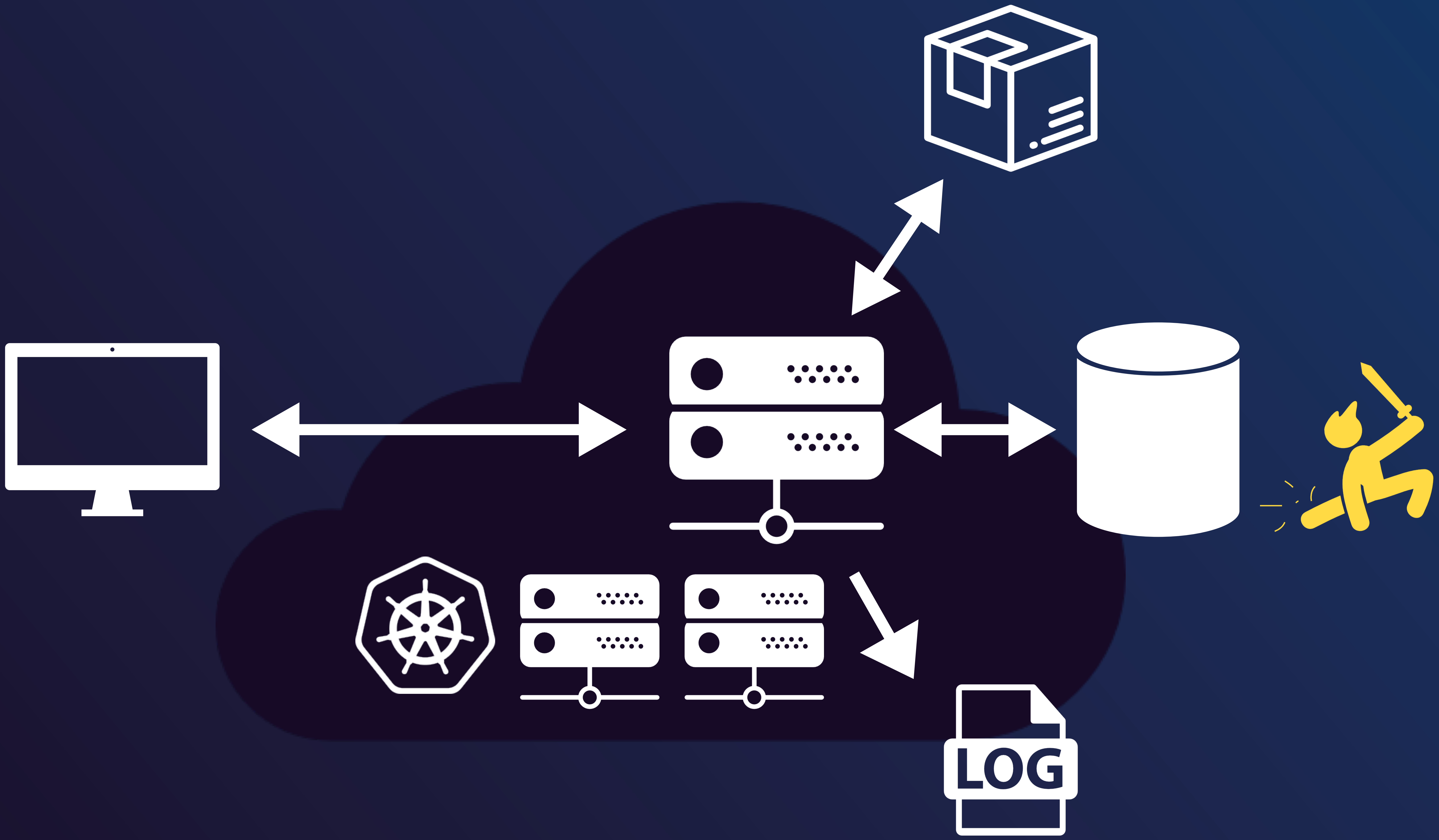
@vixentael



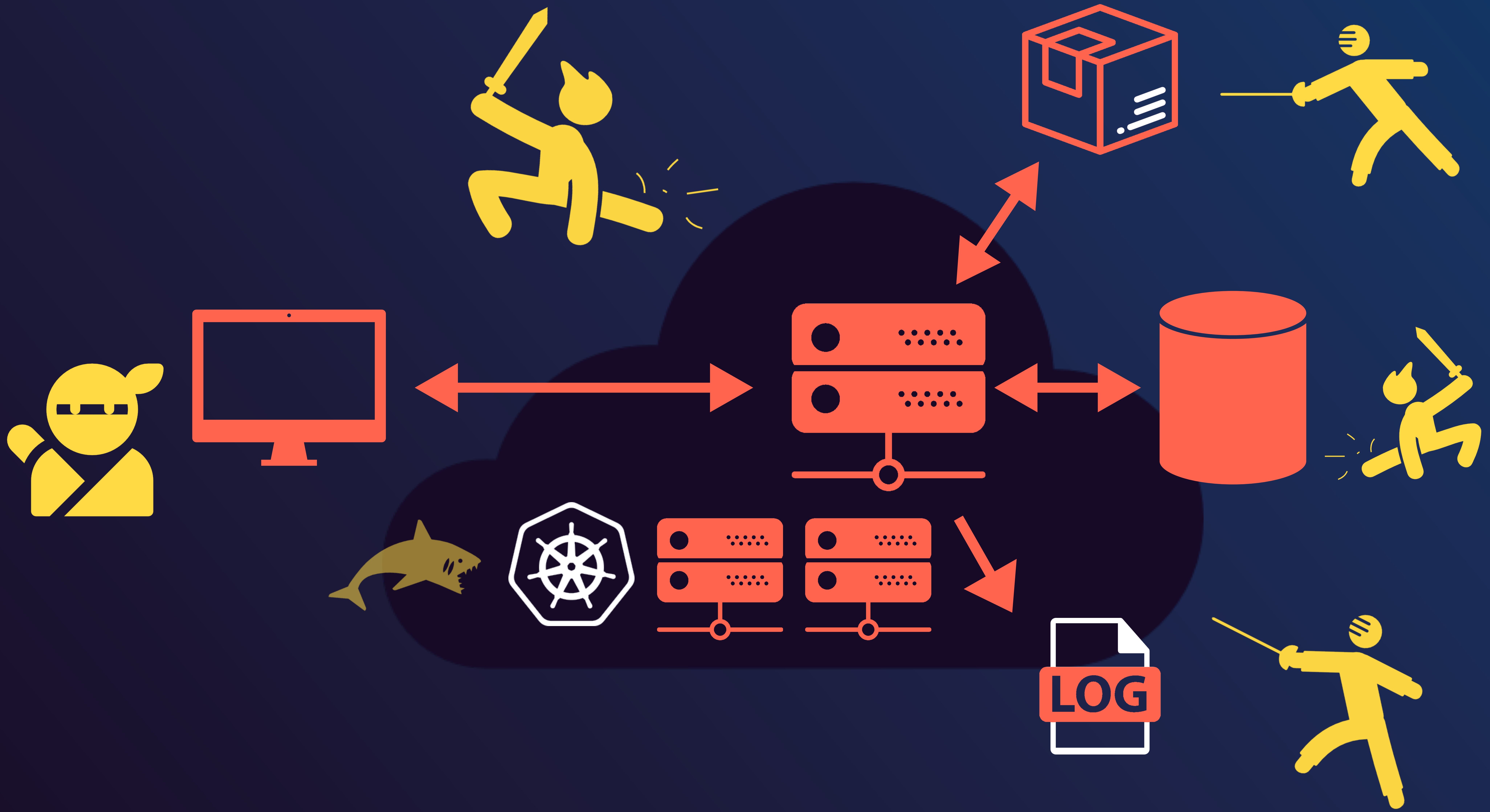
@vixentael



@vixentael

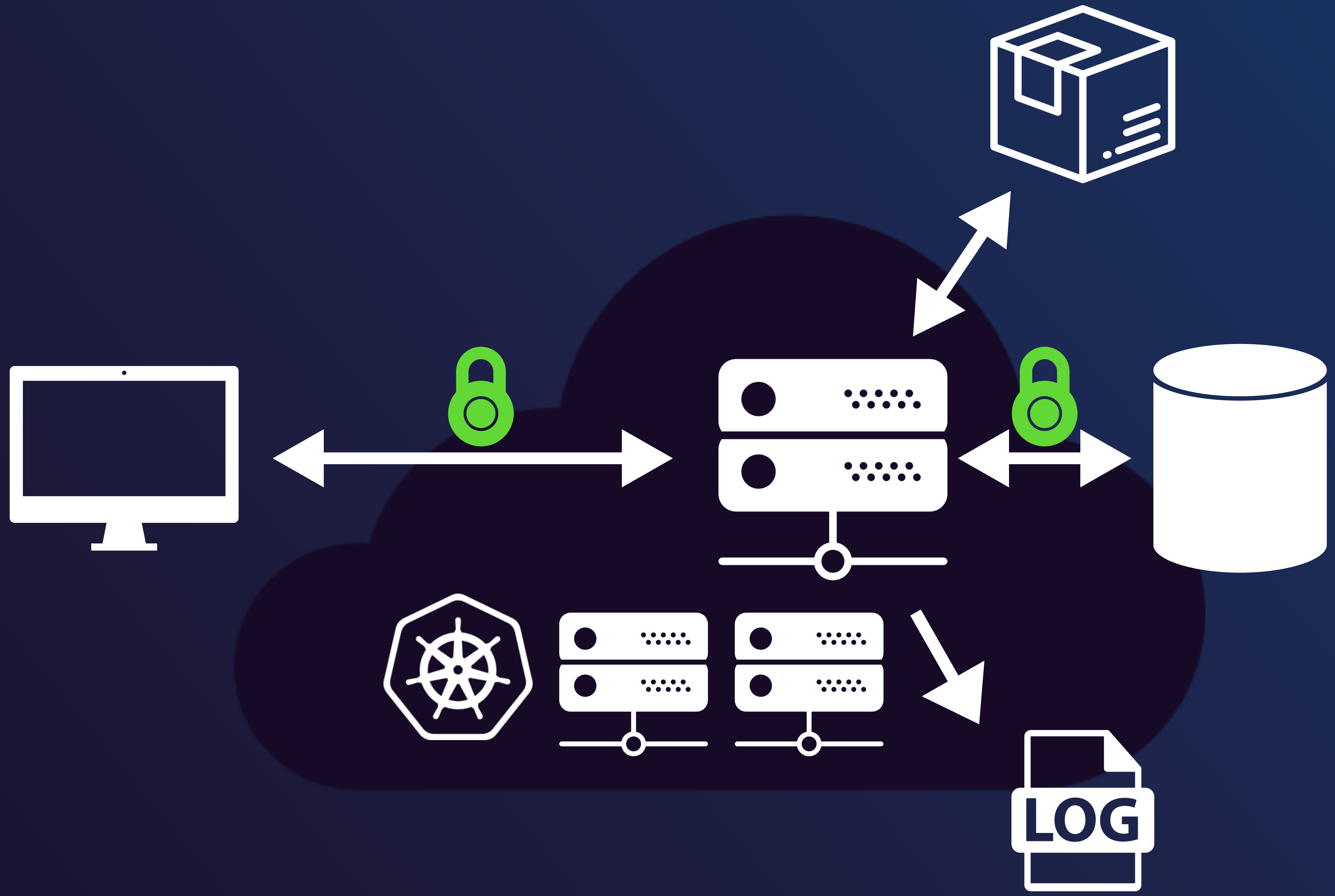


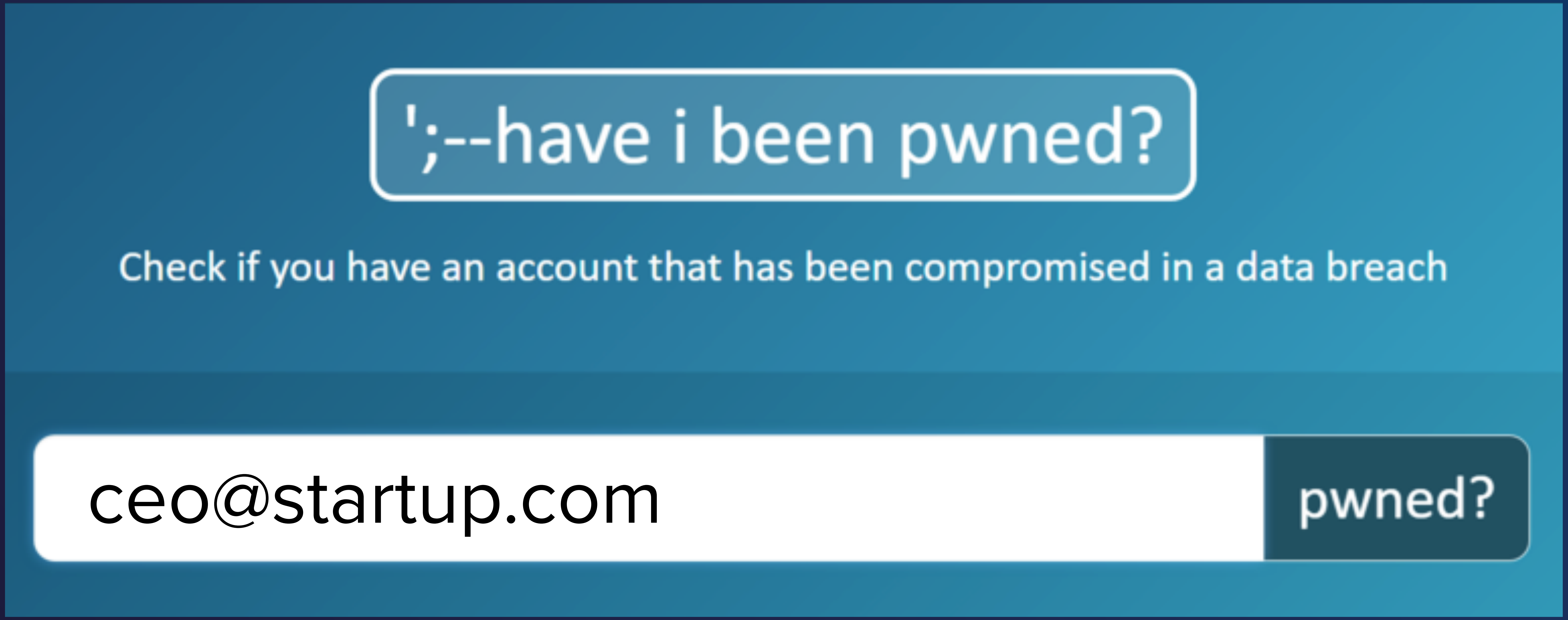
@vixentael



@vixentael

@vixentael

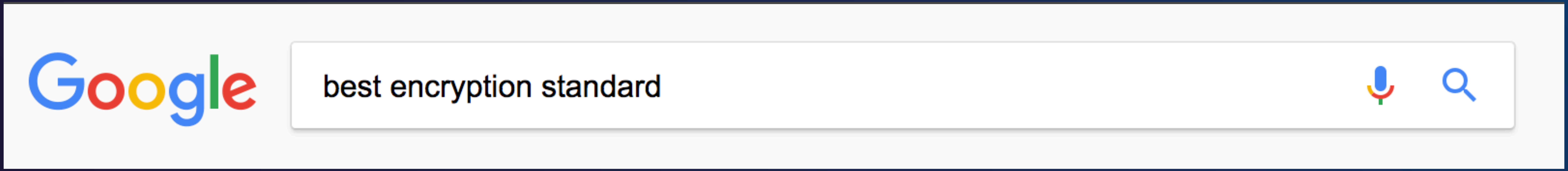

LOG


';--have i been pwned?

Check if you have an account that has been compromised in a data breach

ceo@startup.com

@vixentael



best encryption standard

stackoverflow

stackoverflow

```php
$encrypted= mcrypt_encrypt(
    MCRYPT_RIJNDAEL_128,
    ‘54ca04988748501e93a3061763b0b6a’,
    $data,
    MCRYPT_MODE_CBC,
    $iv
);
```

@vixentael

```
$encrypted= mcrypt_encrypt(
    MCRYPT_RIJNDAEL_128,
   '54ca04988748501e93a3061763b0b6a',
    $data,
    MCRYPT_MODE_CBC,
    $iv
);
```

PHP. AES-CBC

stackoverflow

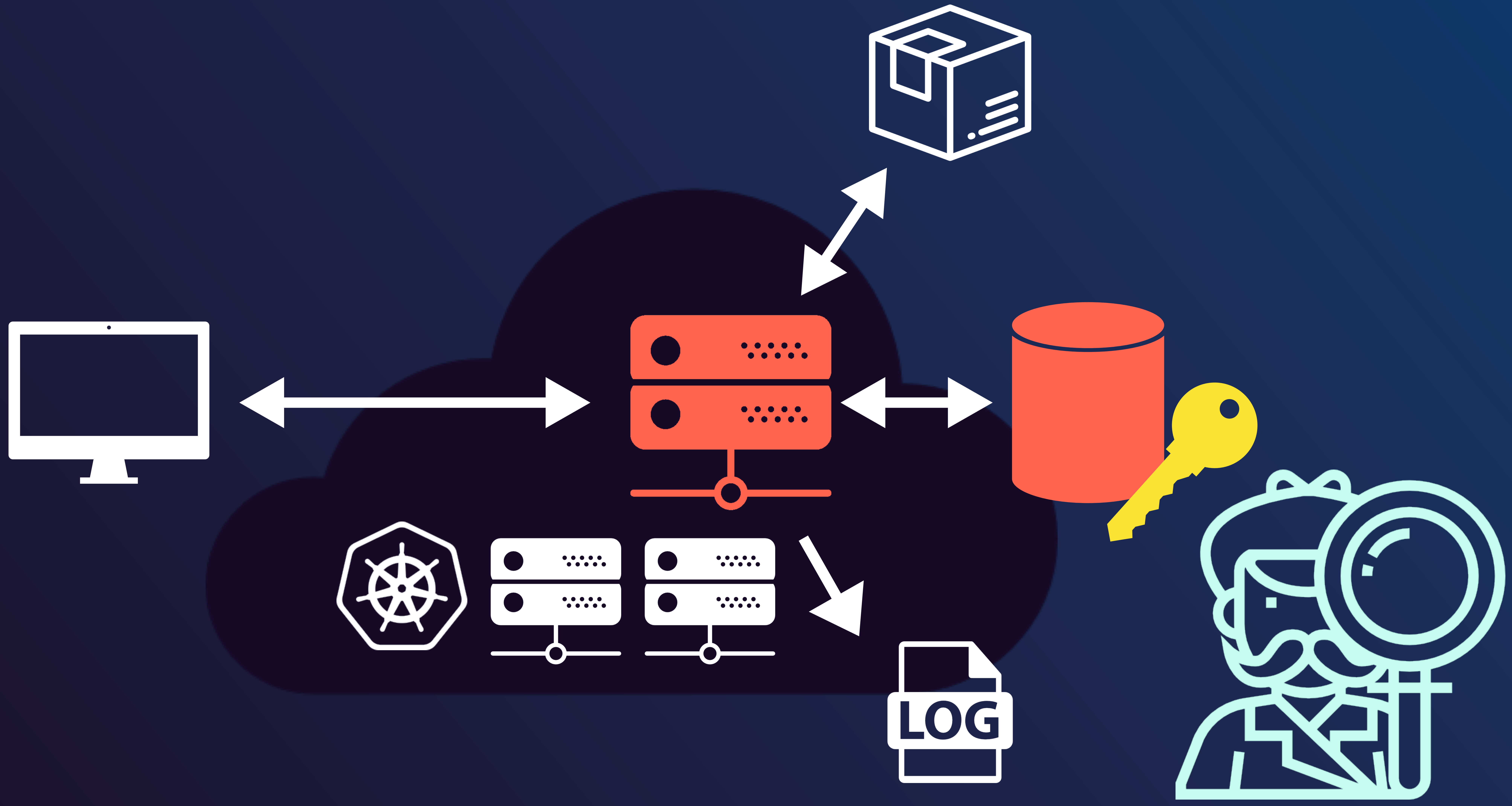
@vixentael

stackoverflow

```
$encrypted= mcrypt_encrypt(
    MCRYPT_RIJNDAEL_128,
    ‘54ca04988748501e93a3061763b0b6a’,
    $data,
    MCRYPT_MODE_CBC,
    $iv
);
```

PHP. AES-CBC

Invite pen-testers!



@vixentael

Use pre-built tools!

@vixentael

```
export PGENCRYPTIONKEY=db-enc-key
initdb -k -K pgcrypto /data/dbencrypt/
```

postgresql encryption options

cybertec-postgresql.com/en/postgresql-instance-level-encryption/

@vixentael

@vixentael

Hire someone?

@vixentael

No data security expertise?
– Find one.

II. The challenge

we want one tool that
solves all problems..

..but how it should work

..and will it really be
secure now?

key lifecycle

trusted code execution environment

side channel resistance

risk echelonization

@vixentael

@vixentael

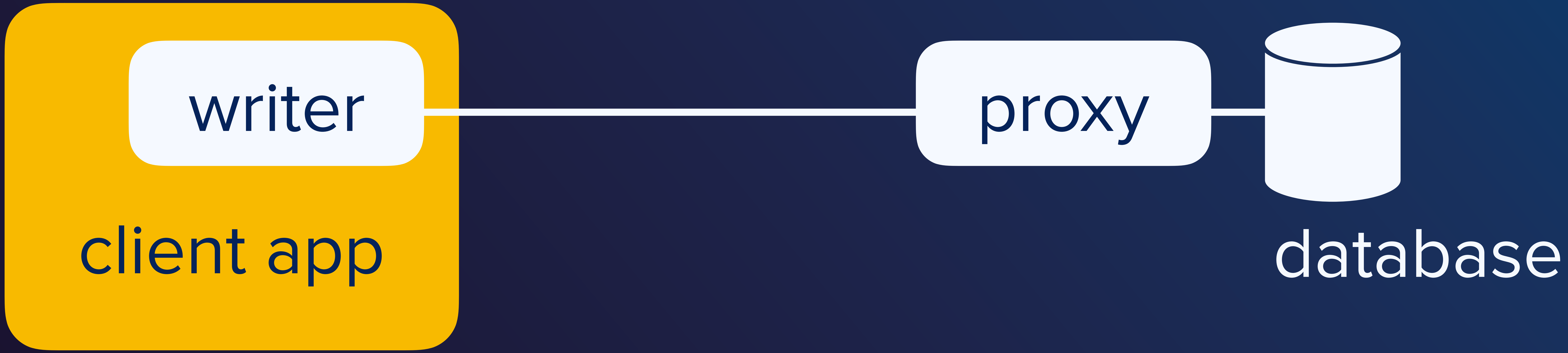
@vixentael

@vixentael

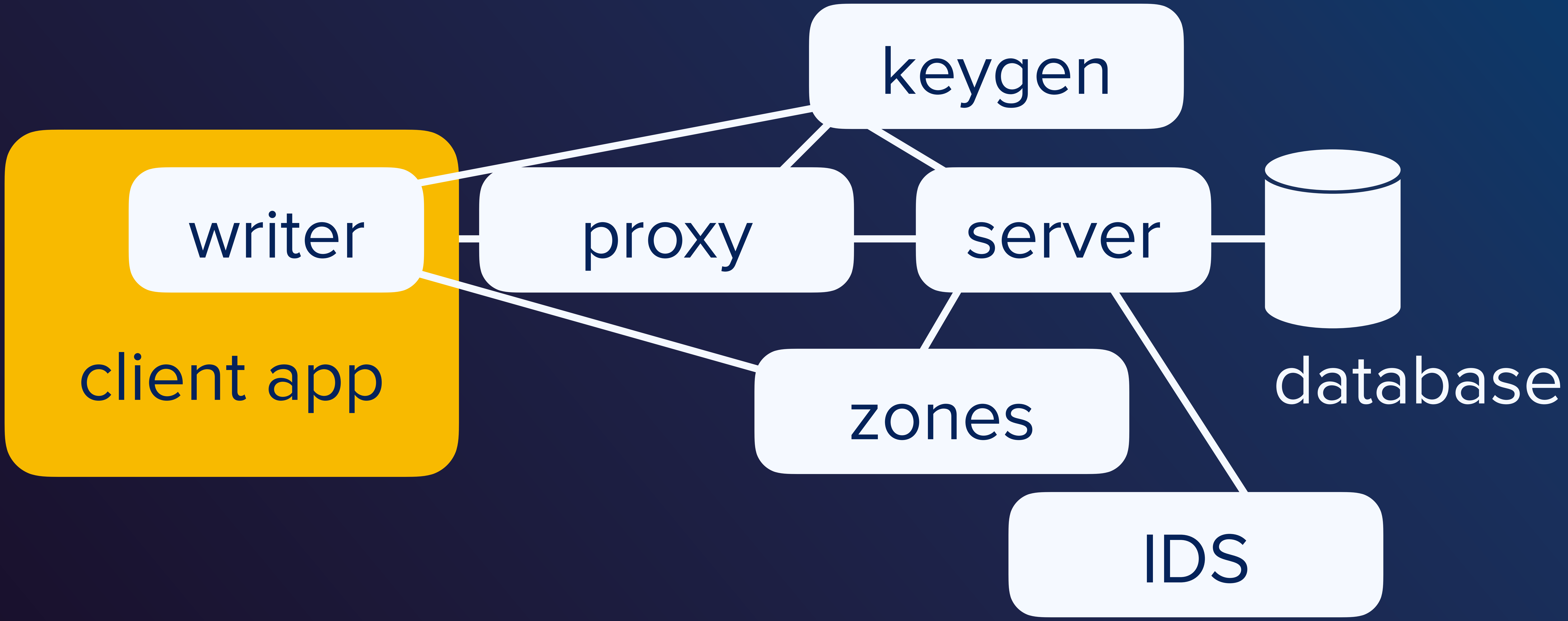
database encryption proxy

@vixentael

database encryption proxy

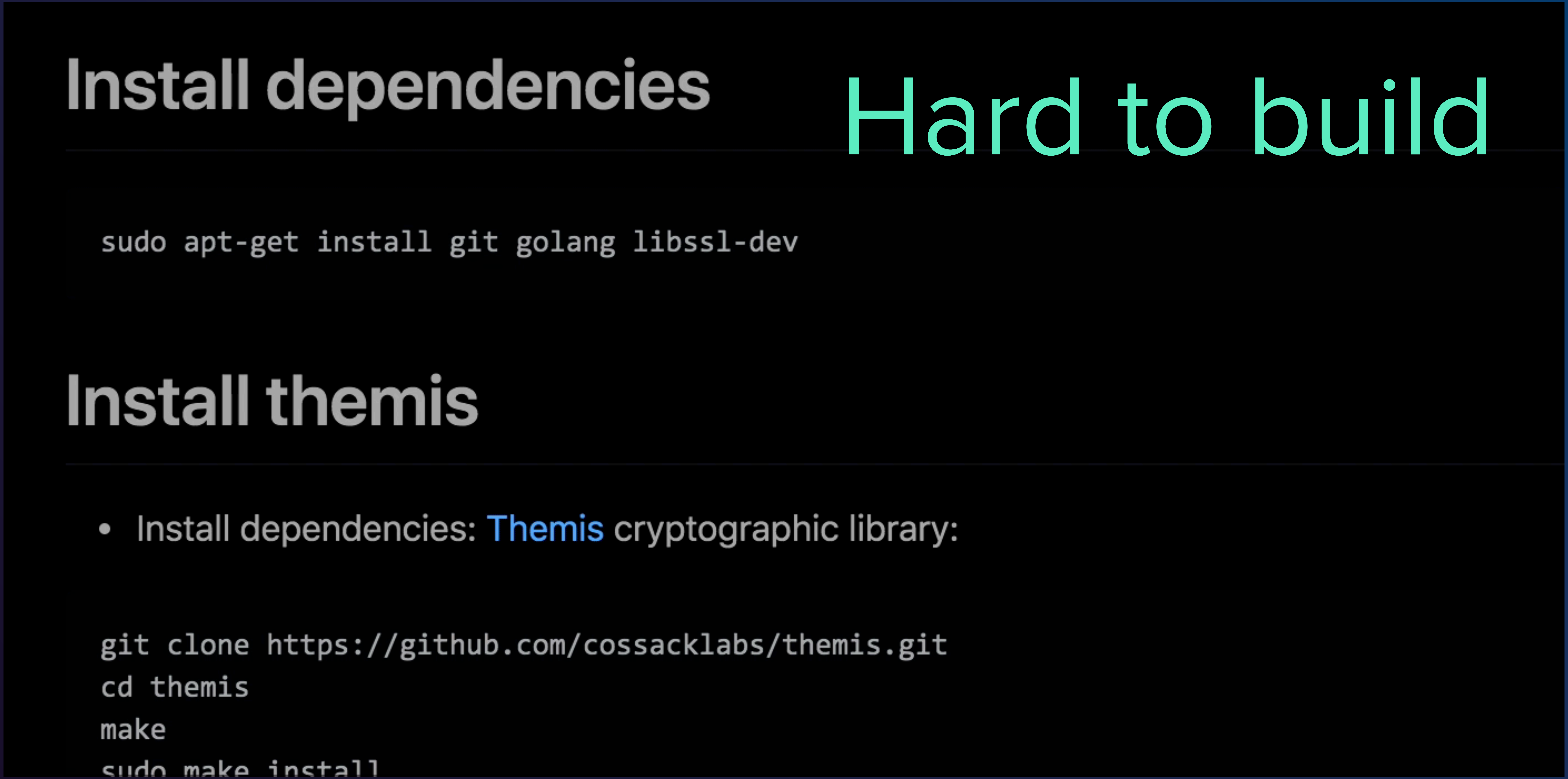


@vixentael



@vixentael

Hard to build



Install dependencies

```
sudo apt-get install git golang libssl-dev
```

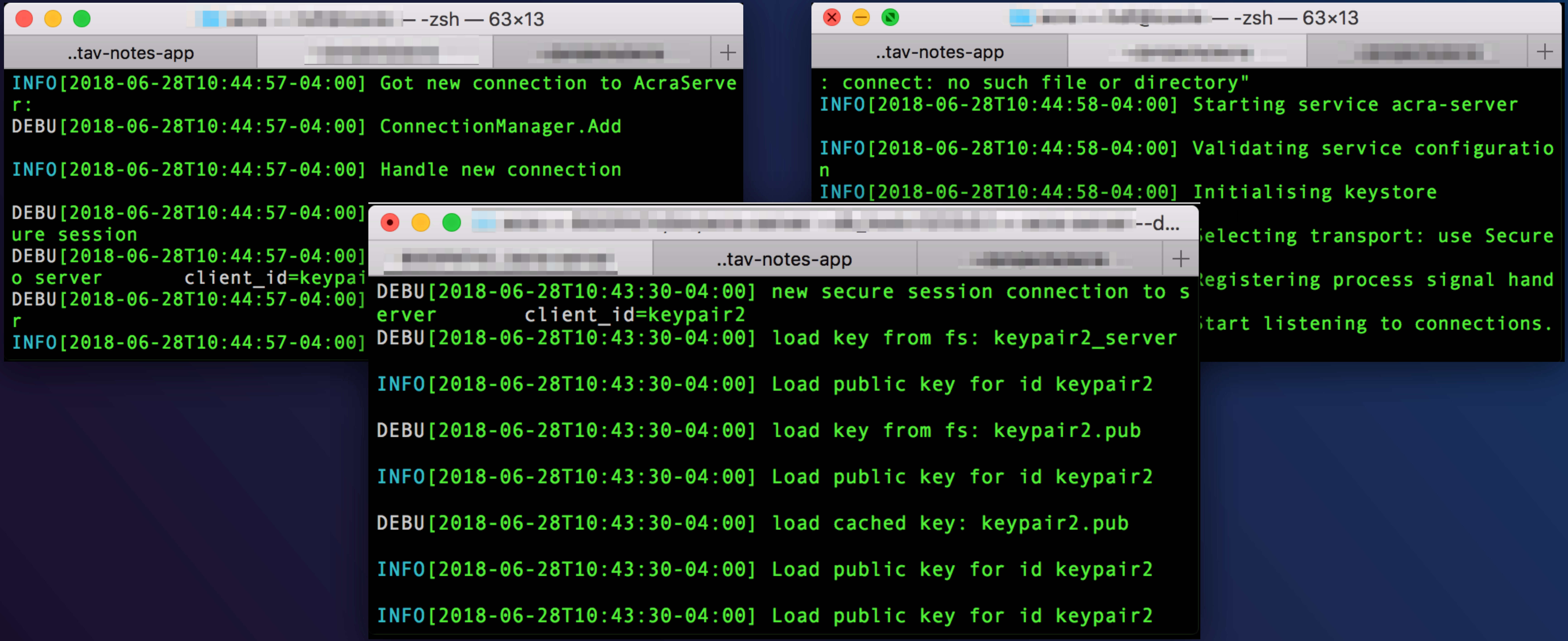
Install themis

- Install dependencies: Themis cryptographic library:

```
git clone https://github.com/cossacklabs/themis.git
cd themis
make
sudo make install
```

@vixentael

Pain to manage



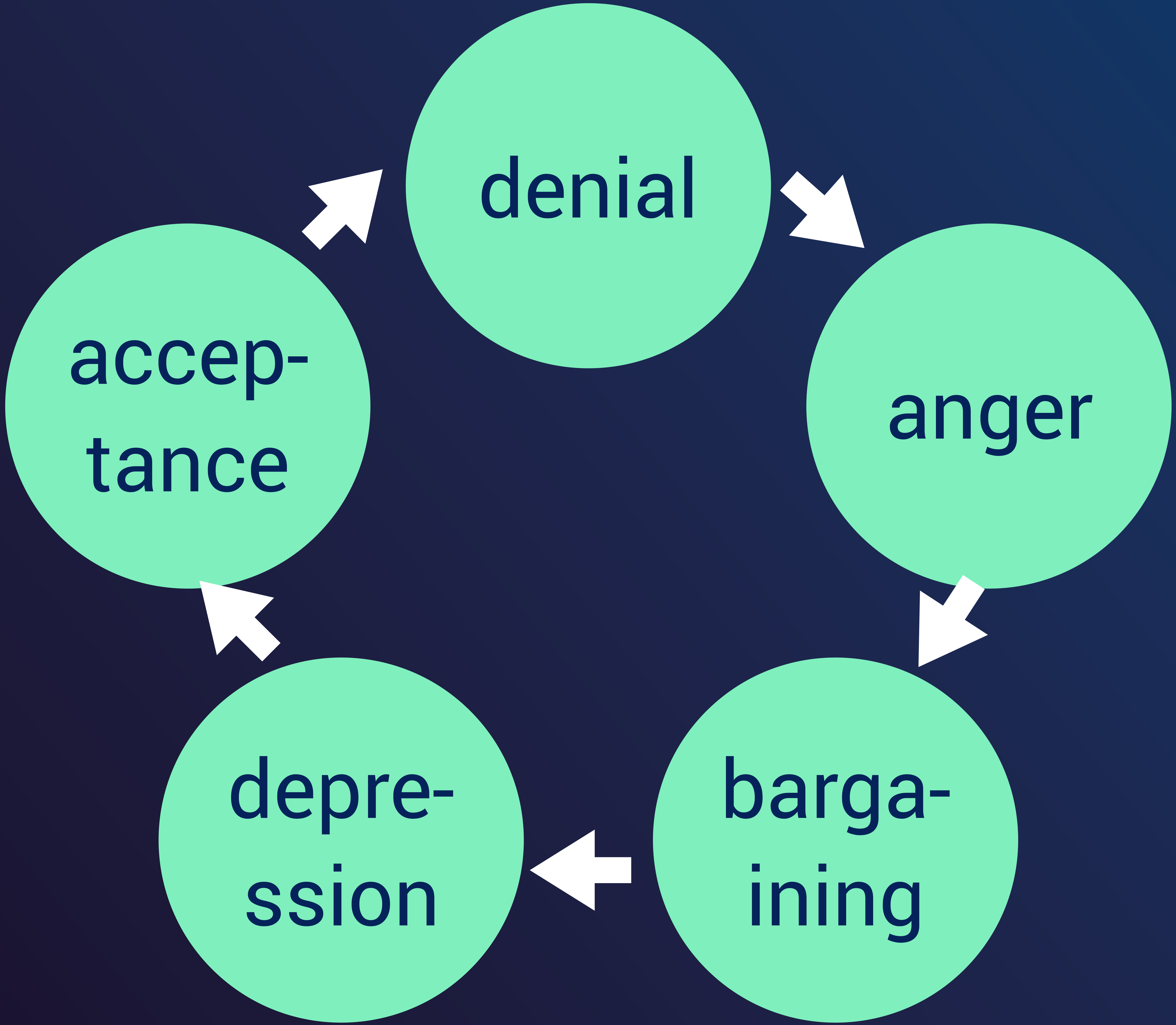
```
INFO[2018-06-28T10:44:57-04:00] Got new connection to AcraServe
r:
DEBU[2018-06-28T10:44:57-04:00] ConnectionManager.Add

INFO[2018-06-28T10:44:57-04:00] Handle new connection

DEBU[2018-06-28T10:44:57-04:00]
ure session
DEBU[2018-06-28T10:44:57-04:00]
o server        client_id=keypai
DEBU[2018-06-28T10:44:57-04:00]
r
INFO[2018-06-28T10:44:57-04:00]
```

```
: connect: no such file or directory"
INFO[2018-06-28T10:44:58-04:00] Starting service acra-server

INFO[2018-06-28T10:44:58-04:00] Validating service configuratio
n
INFO[2018-06-28T10:44:58-04:00] Initialising keystore

                               electing transport: use Secure

                               egistering process signal hand

                               tart listening to connections.
```

```
DEBU[2018-06-28T10:43:30-04:00] new secure session connection to s
erver        client_id=keypair2
DEBU[2018-06-28T10:43:30-04:00] load key from fs: keypair2_server

INFO[2018-06-28T10:43:30-04:00] Load public key for id keypair2

DEBU[2018-06-28T10:43:30-04:00] load key from fs: keypair2.pub

INFO[2018-06-28T10:43:30-04:00] Load public key for id keypair2

DEBU[2018-06-28T10:43:30-04:00] load cached key: keypair2.pub

INFO[2018-06-28T10:43:30-04:00] Load public key for id keypair2

INFO[2018-06-28T10:43:30-04:00] Load public key for id keypair2
```

@vixentael

@vixentael



@vixentael

Listen to customers.
It improves everything…
even security!

III. The adventure

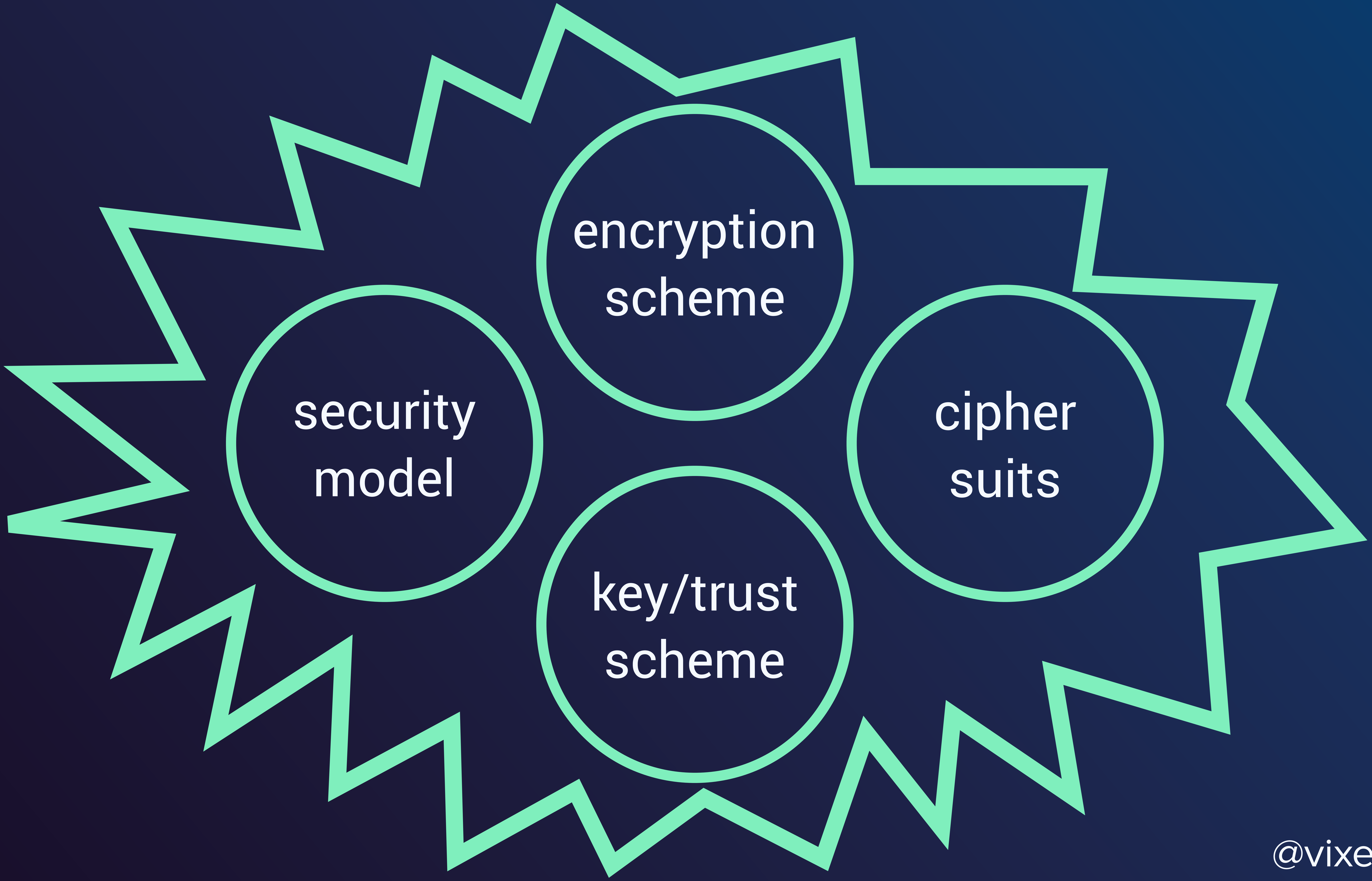
encryption scheme

security model

cipher suits

key/trust scheme

@vixentael


encryption scheme
security model
cipher suits
key/trust scheme

@vixentael

– real time analytics (user actions)

– servers load

– error logs

– open tickets / issues

– user testing / user research

- ~~real time analytics (user actions)~~
- ~~servers load~~
- ~~error logs~~
- open tickets / issues
- user testing / user research

@vixentael

Bad Usability → Bad Security

el

Data Security Assistance Program

- business model / regulations
- risks to data
- threat model / attack vectors
- data security scheme

@vixentael

Analyze use-cases

@vixentael

Analyze use-cases

Easy to misuse

Hard to deploy

Hard to verify

Hard to support

@vixentael

@vixentael

Deployment

@vixentael

Deployment

Multiple channels of distribution

code

@vixentael

Deployment

Multiple channels of distribution

code

Deployment

Multiple channels of distribution

code

built packages (.pkg)

@vixentael

Deployment

Multiple channels of distribution

- code
- built packages (.pkg)
- docker images
- docker compose
- chef configuration
- VM images

@vixentael

Deployment

@vixentael

Deployment

1. Download, build, install every component
2. Generate keys / tokens for each component
3. Put keys into right folders (PK exchange)
4. Configure each component (port, keys)
5. Run components using correct config

@vixentael

Deployment

1. Download, build, install every component
   `script`
2. Generate keys / tokens for each component
3. Put keys into right folders (PK exchange)
4. Configure each component (port, keys)
5. Run components using correct config

@vixentael

Deployment

1. Download, build, install every component

`script`

2. Generate keys / tokens for each component
3. Put keys into right folders (PK exchange)
4. Configure each component (port, keys)
5. Run components using correct config

@vixentael

Deployment

1. Download, build, install every component

script

2. Generate keys / tokens for each component

3. Put keys into right folders (PK exchange)

4. Configure each component (port, keys)

defaults

5. Run components using correct config

@vixentael

Deployment

`one command!`

1. Download, build, install every component
2. Generate keys / tokens for each component
3. Put keys into right folders (PK exchange)
4. Configure each component (port, keys)
5. Run components using correct config

@vixentael

Deployment

Pre-baked configurations

```
docker-compose -f <compose_file>.yml up
```

Deployment

Pre-baked configurations

`mysql-ssl-server-ssl.yml`

```
MySQL <—SSL—> AServer <—SSL—> client
```

Deployment

Pre-baked configurations

mysql-ssl-server-ssl.yml

```
MySQL <—SSL—> AServer <—SSL—> client
```

pgsql-nossl-server-ssession-connector.yml

```
PostgreSQL <—> AServer <—SecureSession—> AConnector <———> client
                                                      ‘—> AWebconfig
```

@vixentael

Deployment

Pre-baked configurations

Deployment

Integration tests everywhere 🙄

– run on empty environments

– run on 12 OSs

– provide testing scripts for users

@vixentael

Integration

Good products do not exist in a vacuum

– infrastructure as a code (configs everywhere)

– logging formats (plaintext, json, CEF)

– event formats (unique event codes)

Secure by default

@vixentael

Secure by default

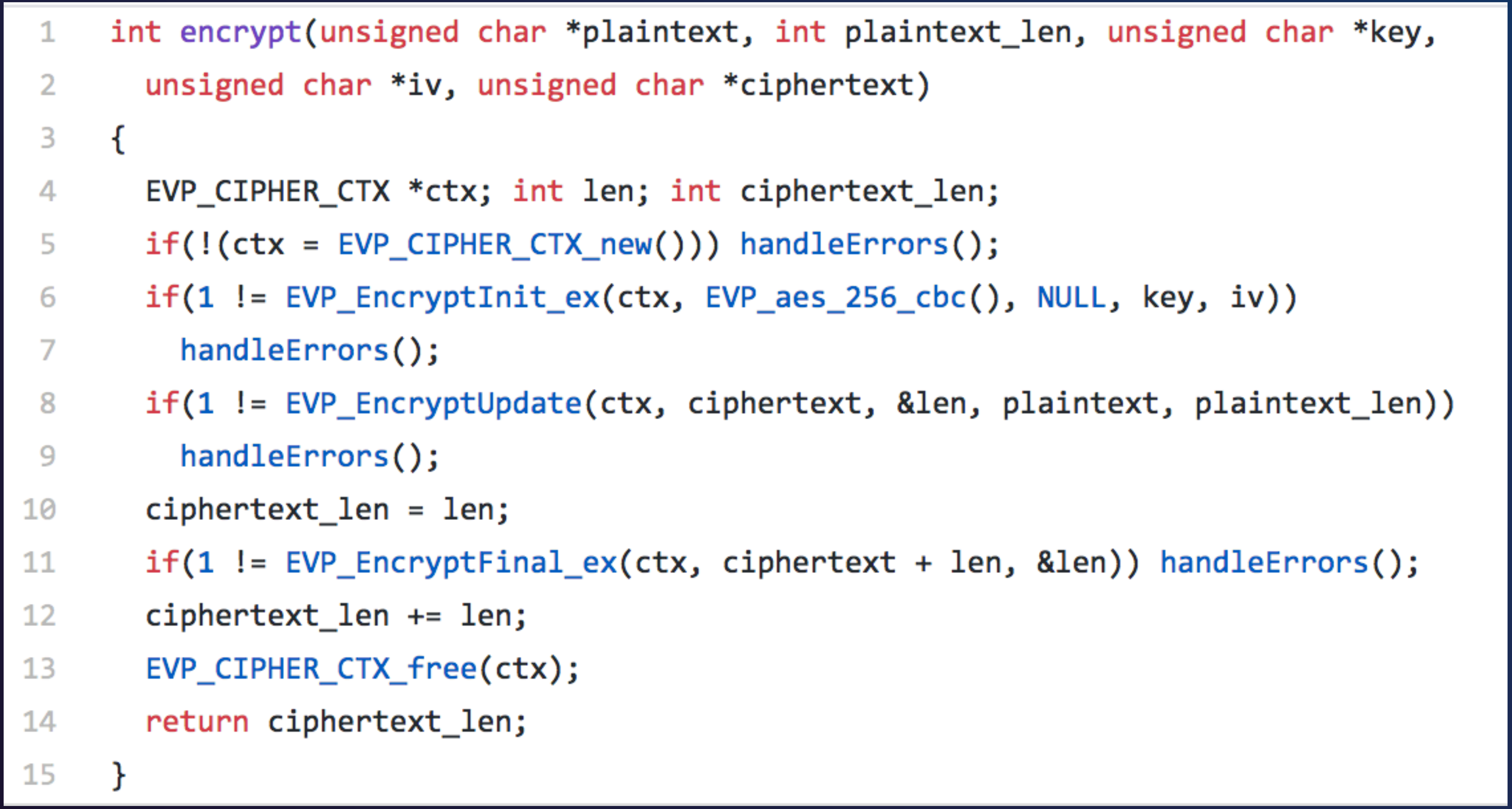
default strict parameters

pre-defined configuration files

make accidental changes unlikely

@vixentael

API design



```c
int encrypt(unsigned char *plaintext, int plaintext_len, unsigned char *key,
  unsigned char *iv, unsigned char *ciphertext)
{
  EVP_CIPHER_CTX *ctx; int len; int ciphertext_len;
  if(!(ctx = EVP_CIPHER_CTX_new())) handleErrors();
  if(1 != EVP_EncryptInit_ex(ctx, EVP_aes_256_cbc(), NULL, key, iv))
    handleErrors();
  if(1 != EVP_EncryptUpdate(ctx, ciphertext, &len, plaintext, plaintext_len))
    handleErrors();
  ciphertext_len = len;
  if(1 != EVP_EncryptFinal_ex(ctx, ciphertext + len, &len)) handleErrors();
  ciphertext_len += len;
  EVP_CIPHER_CTX_free(ctx);
  return ciphertext_len;
}
```

API design

```
from pythemis.scell import SCellSeal

scell = SCellSeal(key)
encrypted_message = scell.encrypt(message, context)


message = scell.decrypt(encrypted_message, context)
```

github.com/cossacklabs

@vixentael

API design

easy to use

&&

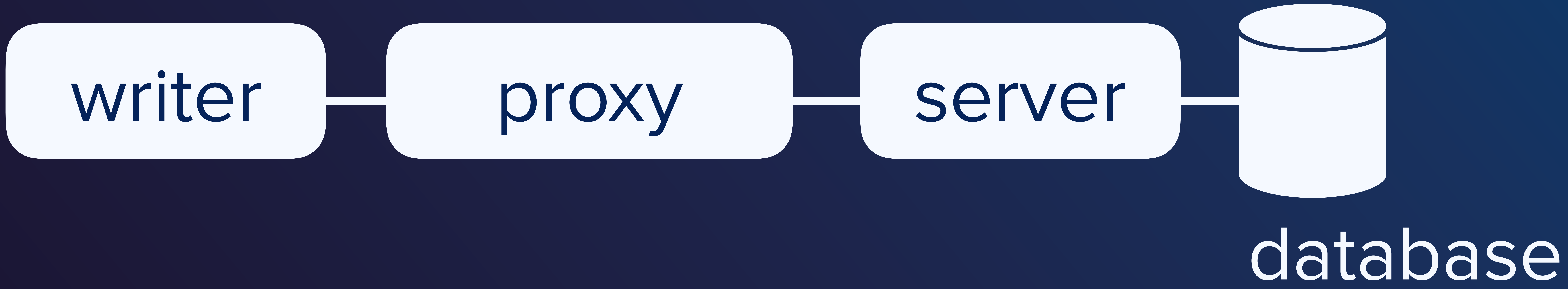
unambiguous to use

2017.hack.lu/archive/2017/hacklu-crypto-api.pdf

Naming

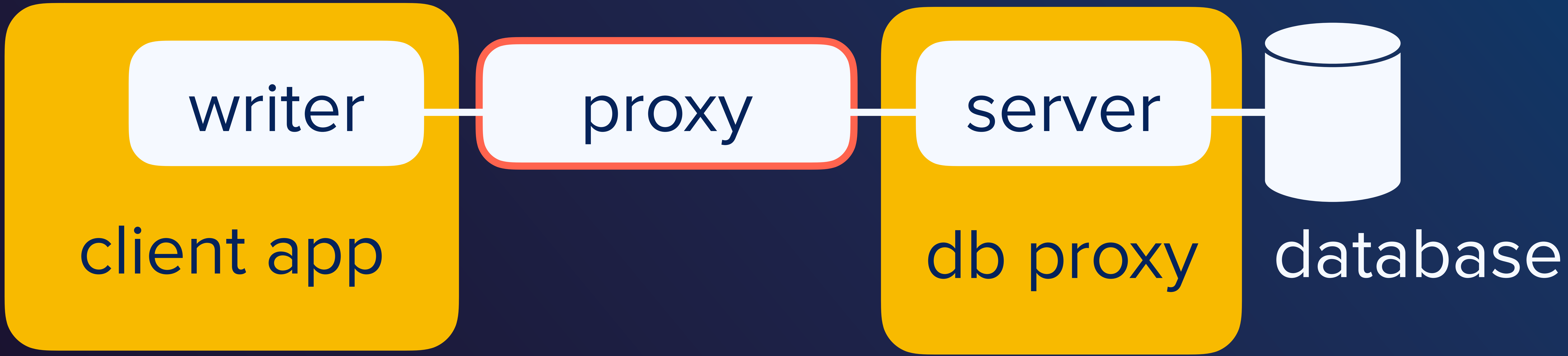
@vixentael

Naming



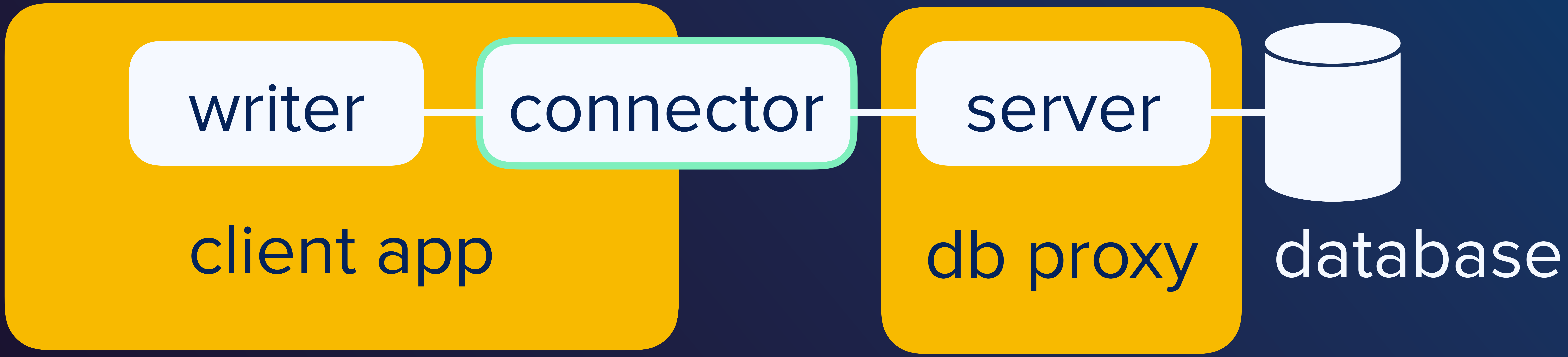
@vixentael

Naming



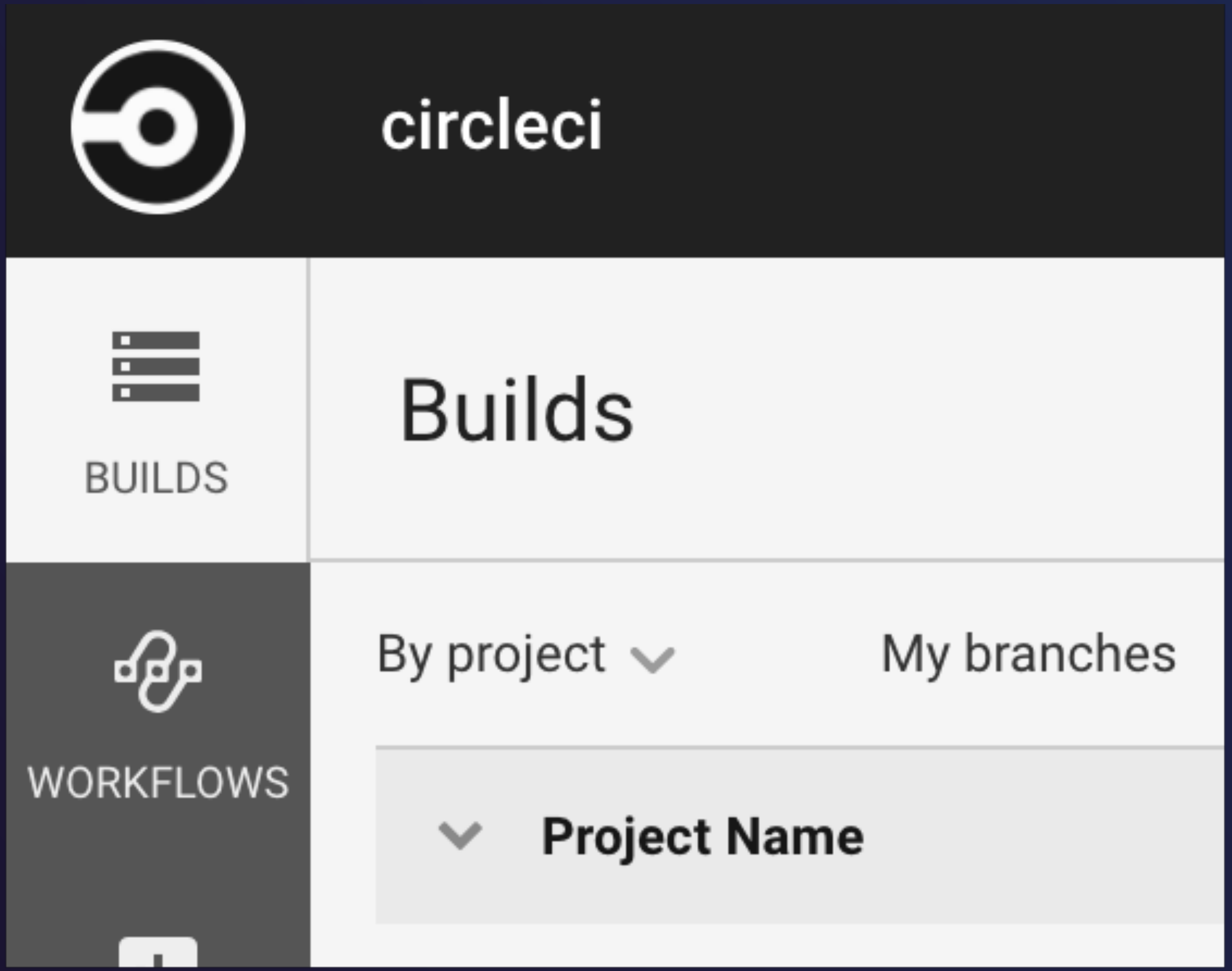
@vixentael

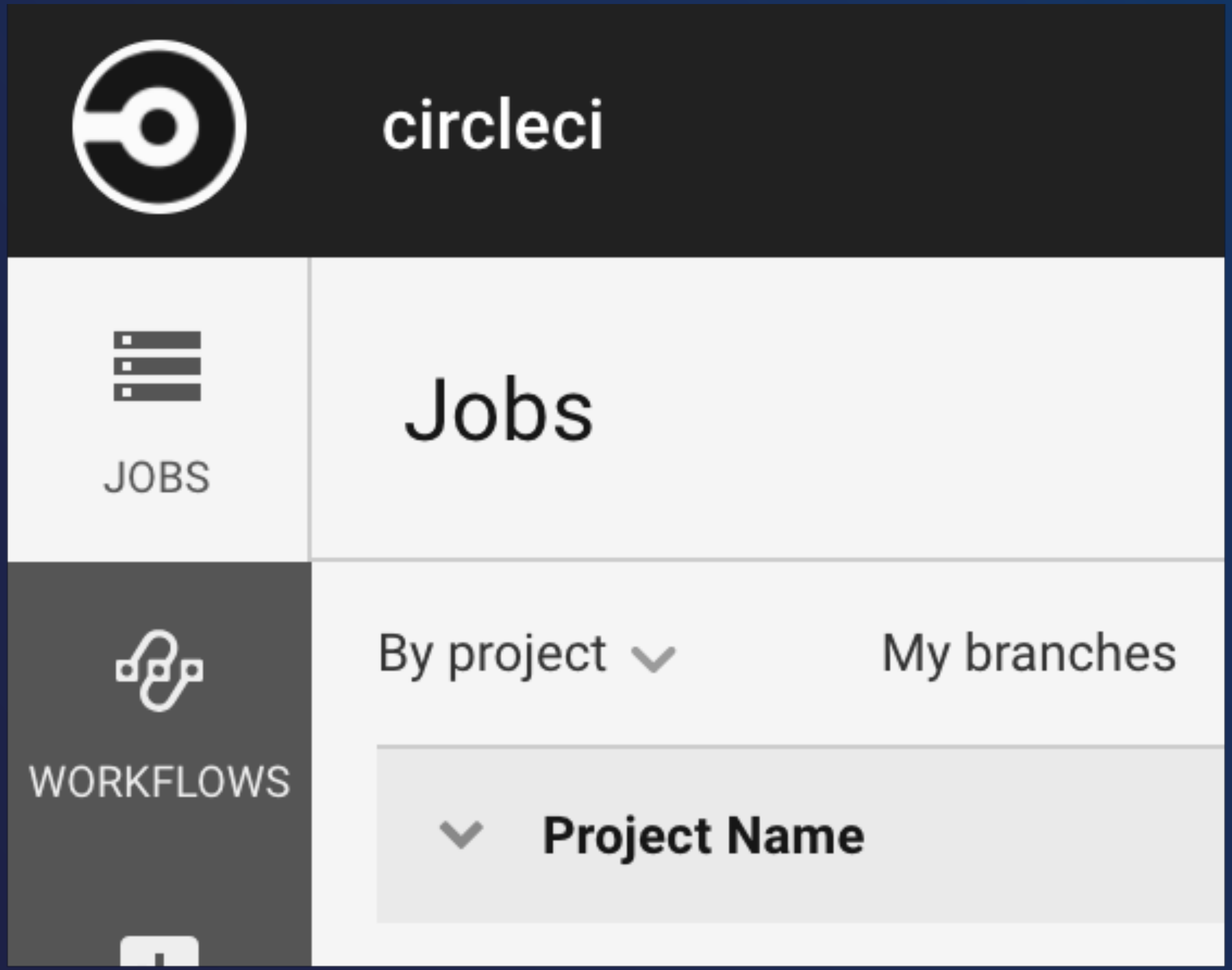
Naming



@vixentael

Naming

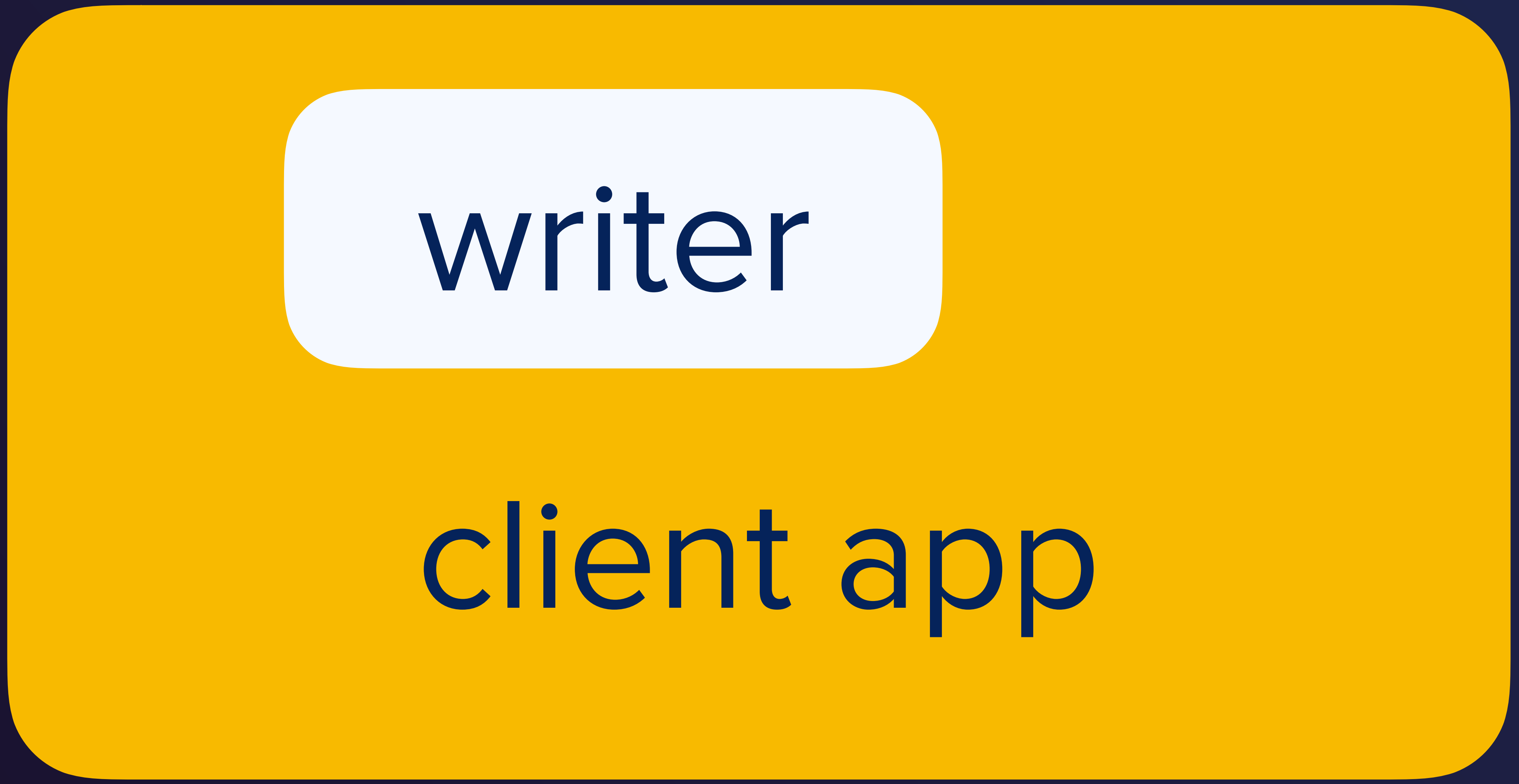




https://circleci.com/blog/why-did-builds-become-jobs-in-the-ui/

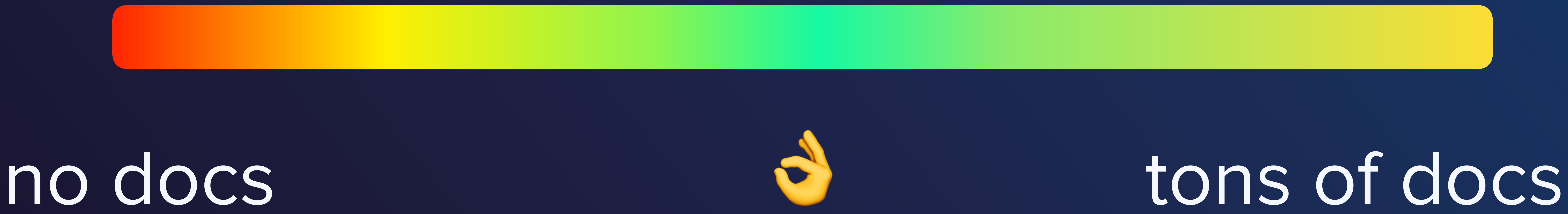
@vixentael

Client side



Go
PHP
Nodejs
Ruby
Python

Docs



@vixentael

Docs

for developers

integration scenarios

security recommendations

simple explanations

benchmarks

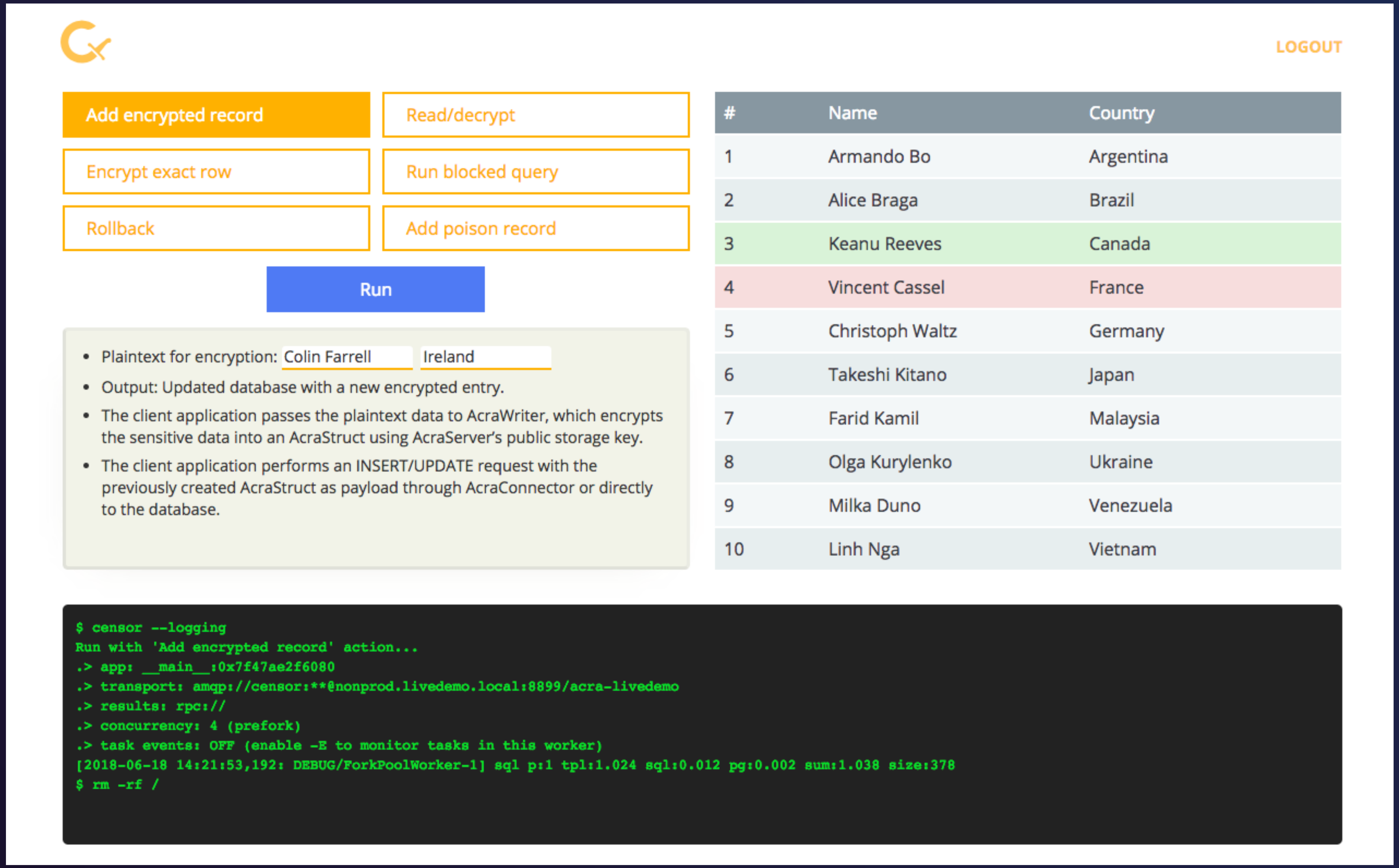
for security ppl

security model

threat vectors

schemes & formulas

Playgrounds



- Plaintext for encryption: Colin Farrell Ireland
- Output: Updated database with a new encrypted entry.
- The client application passes the plaintext data to AcraWriter, which encrypts the sensitive data into an AcraStruct using AcraServer's public storage key.
- The client application performs an INSERT/UPDATE request with the previously created AcraStruct as payload through AcraConnector or directly to the database.

#	Name	Country
1	Armando Bo	Argentina
2	Alice Braga	Brazil
3	Keanu Reeves	Canada
4	Vincent Cassel	France
5	Christoph Waltz	Germany
6	Takeshi Kitano	Japan
7	Farid Kamil	Malaysia
8	Olga Kurylenko	Ukraine
9	Milka Duno	Venezuela
10	Linh Nga	Vietnam

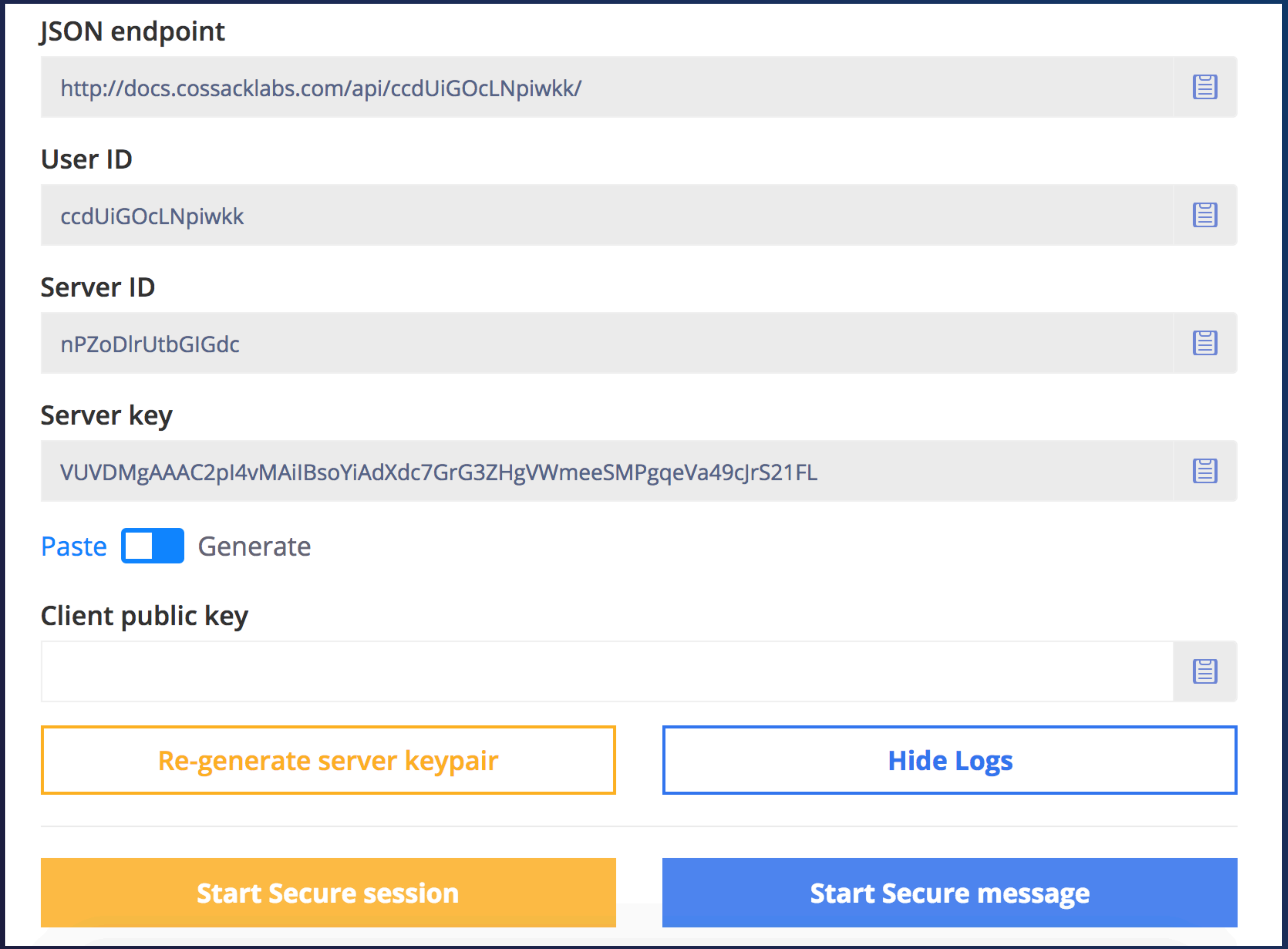
```
$ censor --logging
Run with 'Add encrypted record' action...
.> app: __main__:0x7f47ae2f6080
.> transport: amqp://censor:**@nonprod.livedemo.local:8899/acra-livedemo
.> results: rpc://
.> concurrency: 4 (prefork)
.> task events: OFF (enable -E to monitor tasks in this worker)
[2018-06-18 14:21:53,192: DEBUG/ForkPoolWorker-1] sql p:1 tpl:1.024 sql:0.012 pg:0.002 sum:1.038 size:378
$ rm -rf /
```

who reads docs if you can play with simulator?

@vixentael

Interactive simulator

check your
encryption works



JSON endpoint
http://docs.cossacklabs.com/api/ccdUiGOcLNpiwkk/

User ID
ccdUiGOcLNpiwkk

Server ID
nPZoDlrUtbGIGdc

Server key
VUVDMgAAAC2pl4vMAiIBsoYiAdXdc7GrG3ZHgVWmeeSMPgqeVa49cJrS21FL

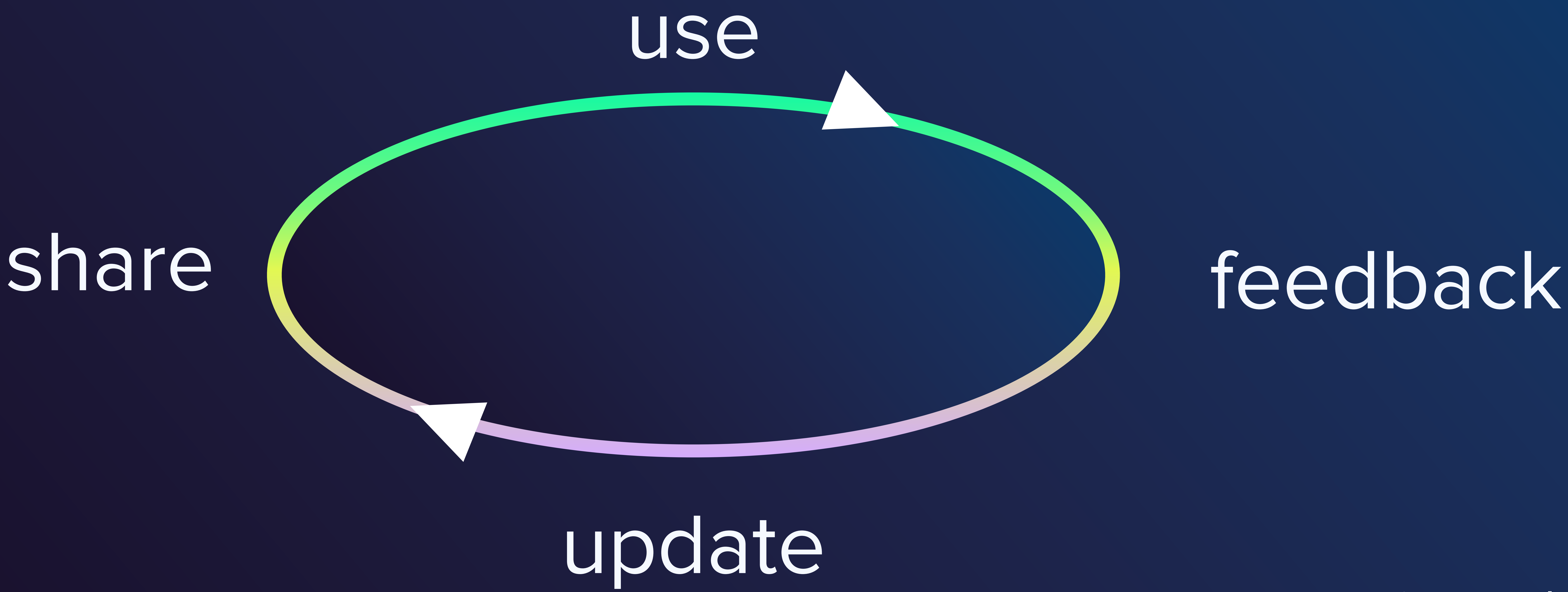
Client public key

@vixentael

Examples-examples-examples

@vixentael

Dogfooding



@vixentael

There is no absolute security

develop test deploy repeat

@vixentael

Short feedback cycle is a key

IV. Where it got us?

Playgrounds

Secure defaults

Unambiguous APIs

Shipped scripts / libs

Easy deployment

Playgrounds

Secu ts

Unambiguous

Shipped scripts / libs

Easy deployment

adopt faster

make less mistakes

become less frustrated

iterate faster

plan better

become less frustrated

make user-facing decisions

usable ≠ over-simplified

Alxbngala

Home reading?

Security as a Product

https://medium.com/@kshortridge/security-as-a-product-83a78c45ca27

Organization security for startups

https://github.com/forter/security-101-for-saas-startups/blob/english/security.md

API design for cryptography

https://2017.hack.lu/archive/2017/hacklu-crypto-api.pdf

Boring crypto, Daniel J. Bernstein

https://cr.yp.to/talks/2015.10.05/slides-djb-20151005-a4.pdf

My other security slides

github.com/vixentael/my-talks

DON'T WASTE TIME ON LEARNING CRYPTOGRAPHY: BETTER USE IT PROPERLY

KEYS FROM THE CASTLE

ANCIENT ART OF MANAGING KEYS AND TRUST

ENCRYPTION WITHOUT MAGIC, RISK MANAGEMENT WITHOUT PAIN

ZERO KNOWLEDGE ARCHITECTURES

for mobile applications

Building user-centric security model in iOS apps

@vixentael

Feel free to reach me with security questions.

I do check my inbox :)

Product Engineer

COSSACK LABS

Image credits

www.flaticon.com

Authors:

freepik, linector, switficons, pixelperfect, smashicons, icon pond, dinosoftlabs