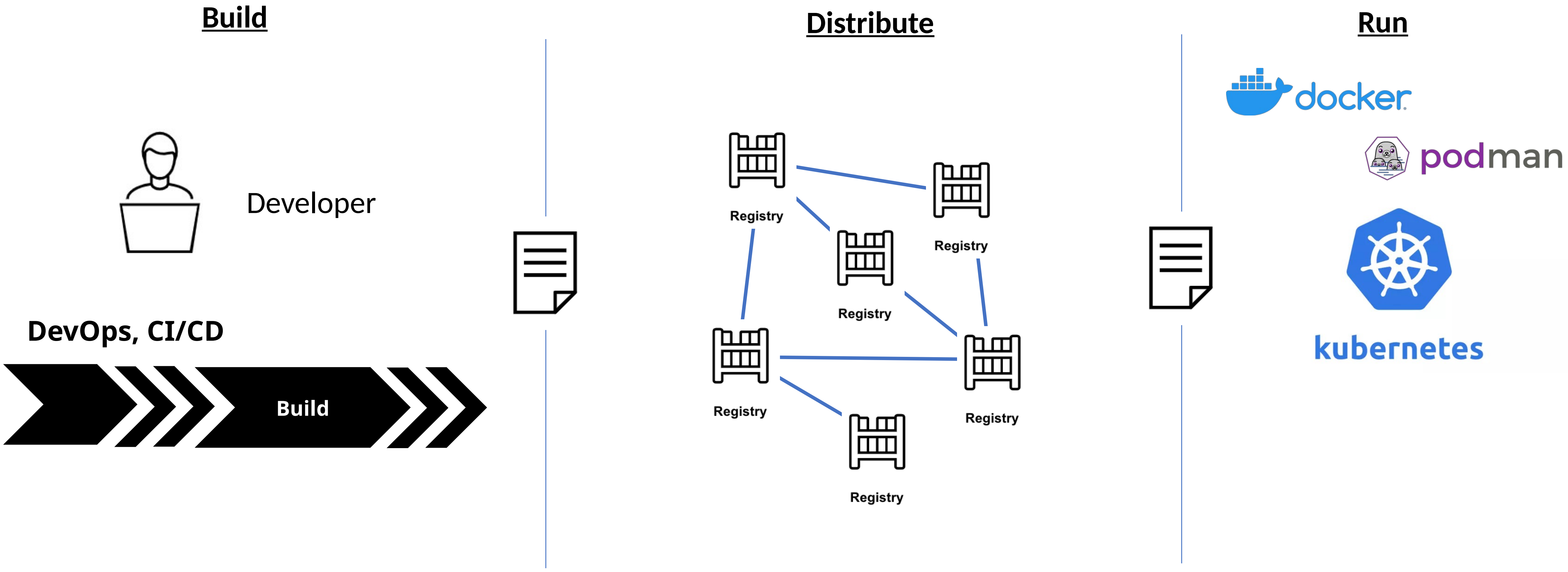
Advancing image security and compliance through Container Image Encryption!

Brandon Lum, IBM

Contents

- Container Image Security
- OCI Container Image Encryption
- End-to-end demo, build, distribute, run!
- Advanced Compliance usecase: Geo-fencing

Container Image Security



Build

Developer

DevOps, CI/CD

Build

Distribute

Registry

Registry

Registry

Registry

Registry

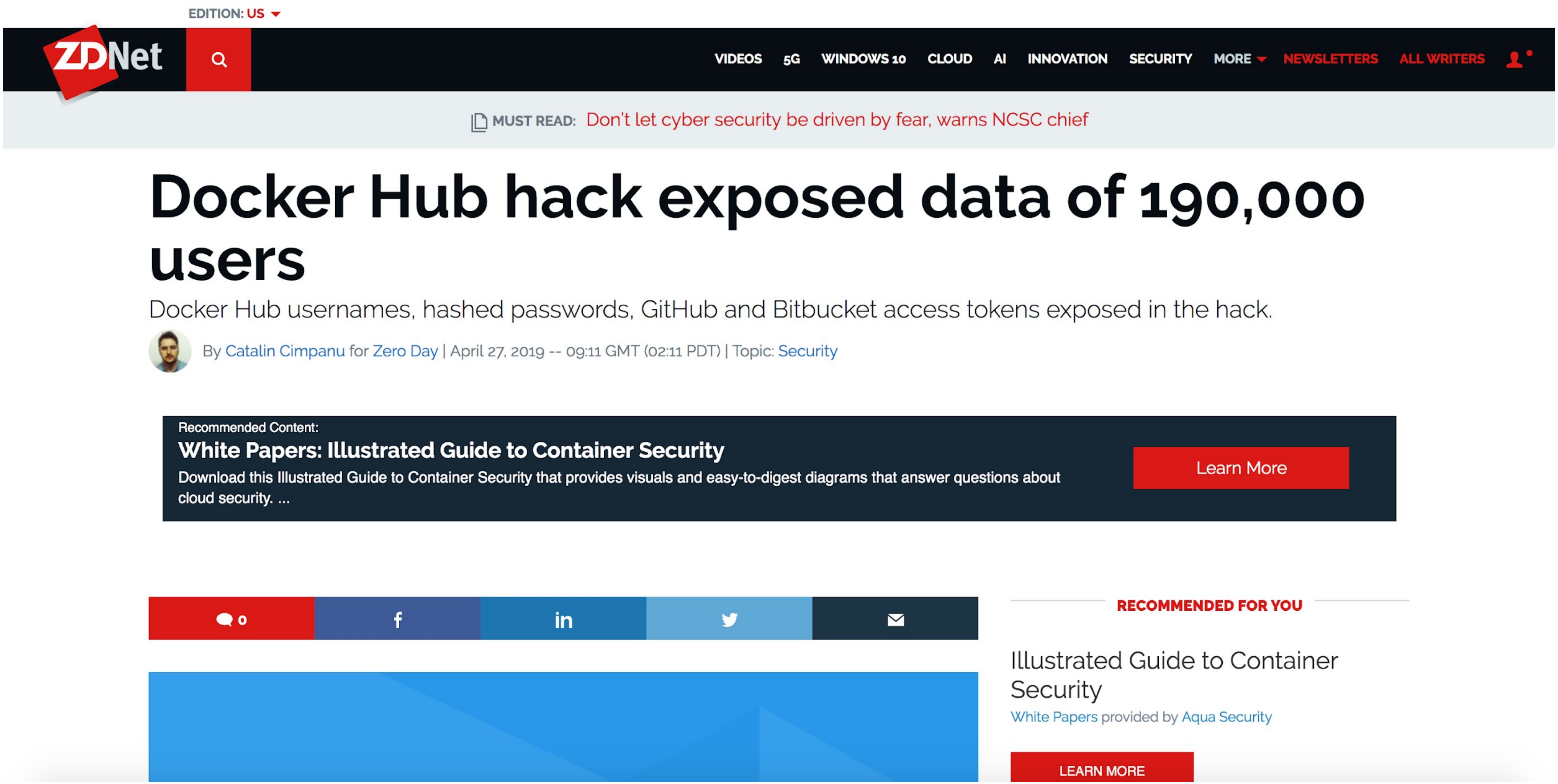
Registry

Run

docker

podman

kubernetes



MUST READ: Don't let cyber security be driven by fear, warns NCSC chief

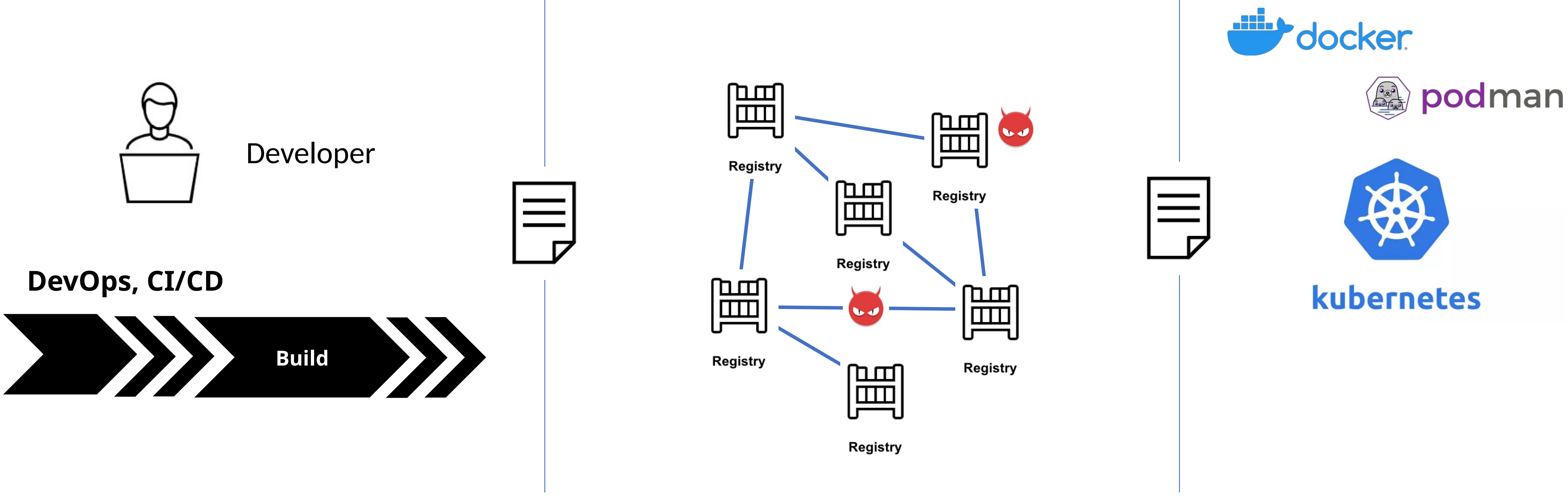
Docker Hub hack exposed data of 190,000 users

Docker Hub usernames, hashed passwords, GitHub and Bitbucket access tokens exposed in the hack.

By Catalin Cimpanu for Zero Day | April 27, 2019 -- 09:11 GMT (02:11 PDT) | Topic: Security

Container Image Security

Build | **Distribute** | **Run**



Developer

DevOps, CI/CD

Build

Registry

Registry

Registry

Registry

Registry

Registry

Registry

- Registries and Cloud may be compromised/untrusted
- Compliance perspective: some of these services in between are not auditable

What does this mean for your images?

notary

PORTIERIS

RedHat Signing

Image Signing will ensure the integrity of your deployment images! *Technologies: Docker Content Trust (DCT) or RedHat Simple Signing*

But…

Private Images' sensitive content will be exposed!

Container Image Encryption

Build

- Build as normal
- **Encrypt**
- Push

Encrypt

- Encrypted image stored
- Cannot be read

Run

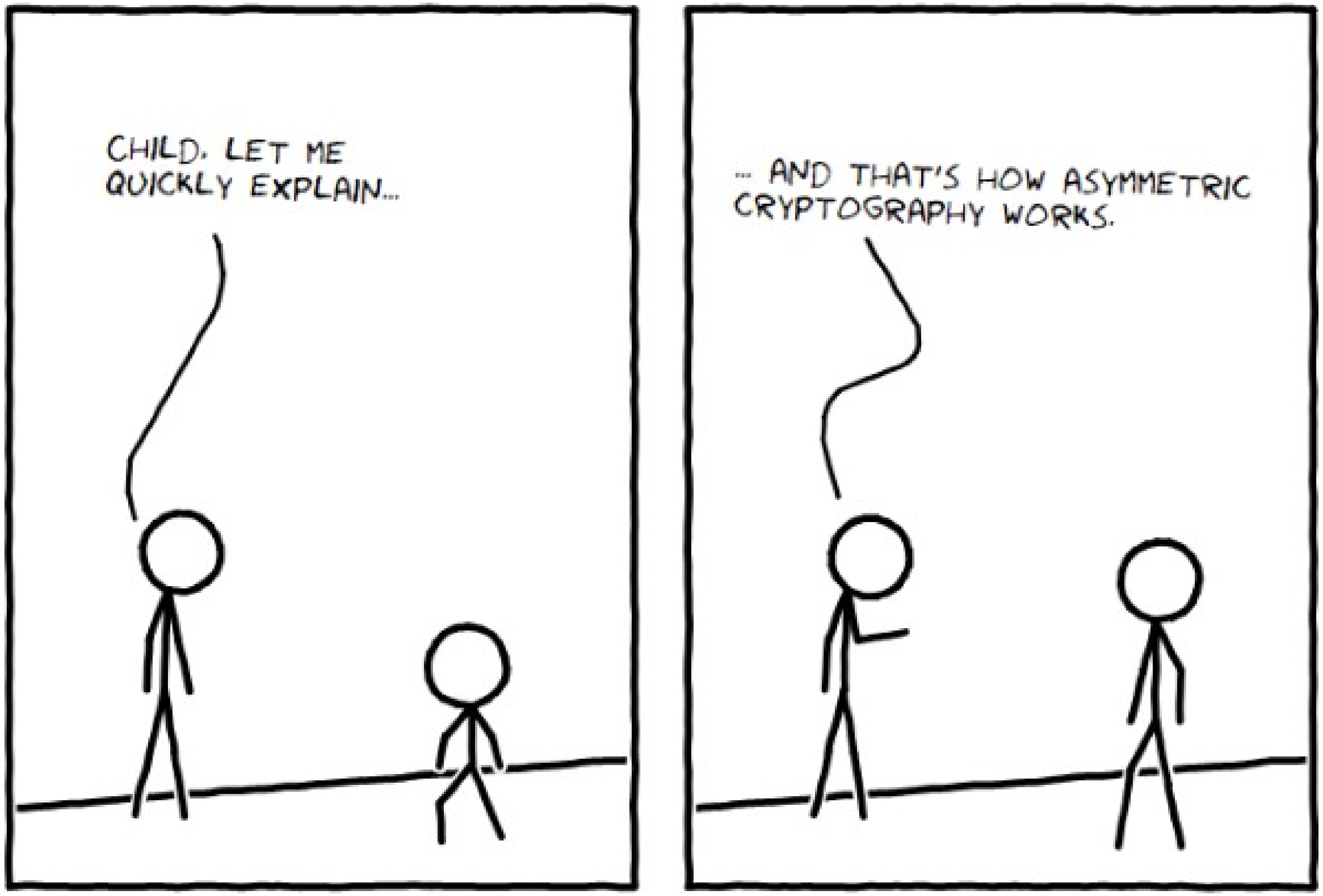
- Pull
- **Decrypt**
- Run

Benefits?

- Image Confidentiality & Deprivileged Registry

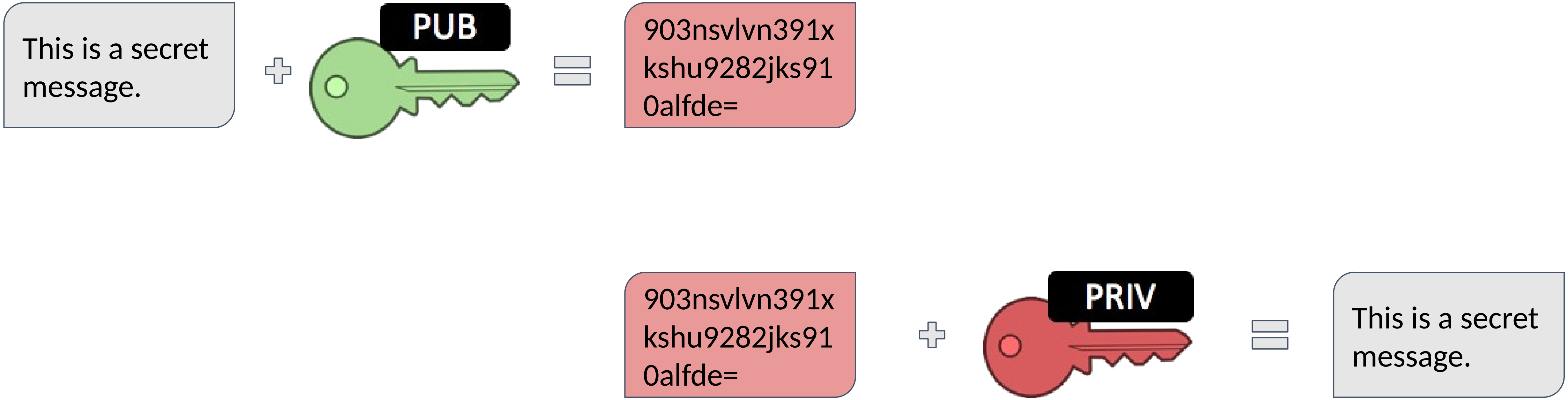
- Execution Boundary Control
  “If my code is running, I know it’s in my cluster”
  - Encrypted Containers Images + Key management could provide guarantees about where an image can run.
  - i.e. Image X can only run in the EU nodes.

Encryption Primer


CHILD, LET ME QUICKLY EXPLAIN...
... AND THAT'S HOW ASYMMETRIC CRYPTOGRAPHY WORKS.

https://www.outsystems.com/blog/posts/how-to-teach-child-about-asymmetric-cryptography/

Encryption Primer - Assym Enc.



This is a secret message.

903nsvlvn391x kshu9282jks91 0alfde=

903nsvlvn391x kshu9282jks91 0alfde=

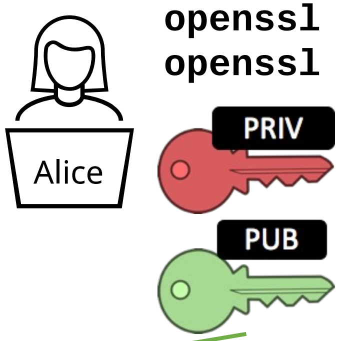
This is a secret message.

Each user has a Public-Private key pair, where Public Key is not secret, can be published.

Bob *sends an encrypted image to* Alice

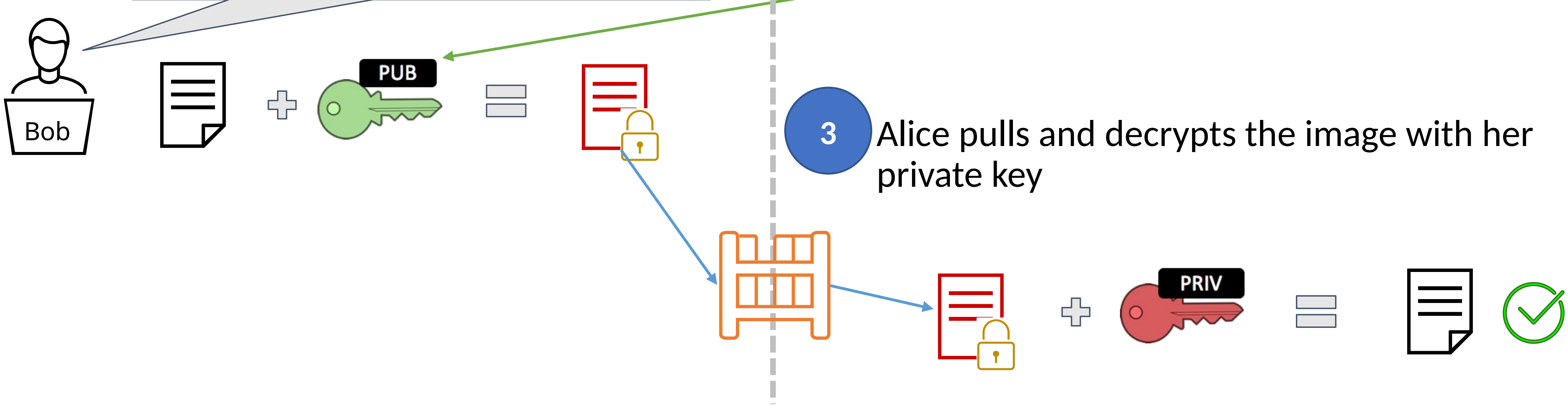
1 Alice creates an RSA private/public keypair, publishing her public key

```
openssl genrsa -out alicePrivate.pem 2048
openssl rsa -in alicePrivate.pem –pubout ...
```



2 Bob encrypts his image and pushes it to the registry

buildah push --encryption- `alicePublic.pem` ...



3 Alice pulls and decrypts the image with her private key

Simple Encrypt/Decrypt Demo

OCI Spec Details

OCI Spec

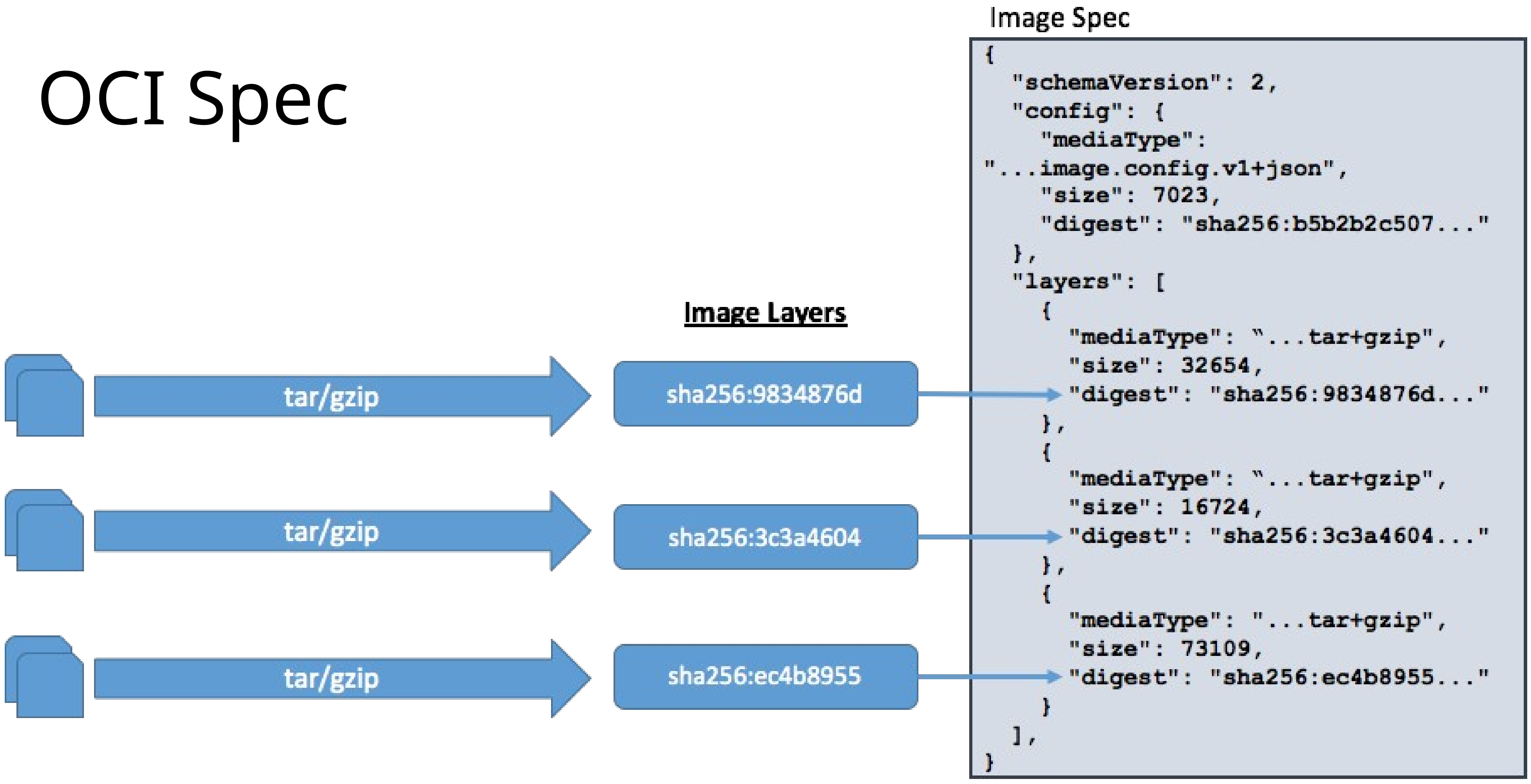


Image Spec

```
{
  "schemaVersion": 2,
  "config": {
    "mediaType":
"...image.config.v1+json",
    "size": 7023,
    "digest": "sha256:b5b2b2c507..."
  },
  "layers": [
    {
      "mediaType": “...tar+gzip",
      "size": 32654,
      "digest": "sha256:9834876d..."
    },
    {
      "mediaType": “...tar+gzip",
      "size": 16724,
      "digest": "sha256:3c3a4604..."
    },
    {
      "mediaType": "...tar+gzip",
      "size": 73109,
      "digest": "sha256:ec4b8955..."
    }
  ],
}
```

OCI Spec **+encrypted**

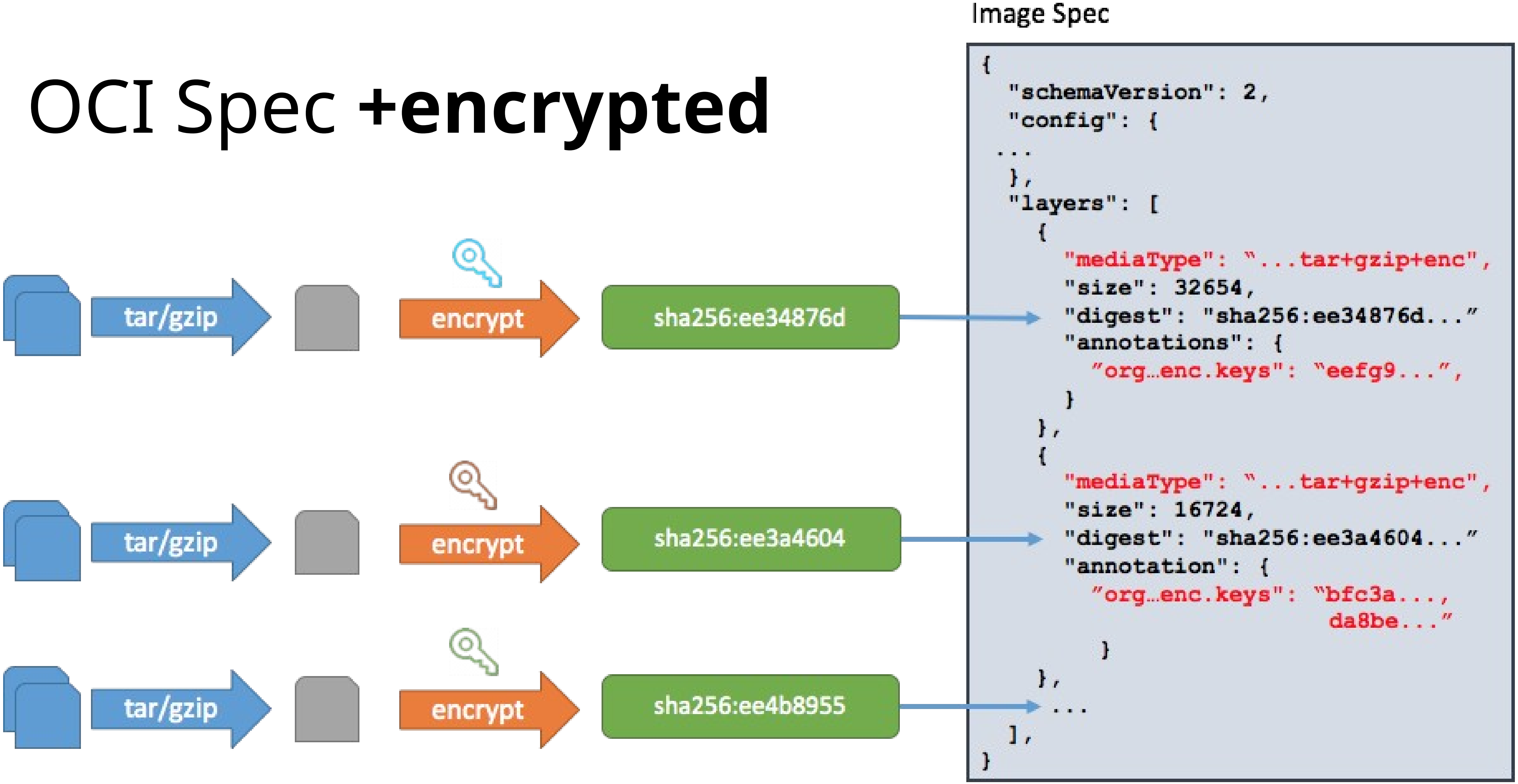


Image Spec

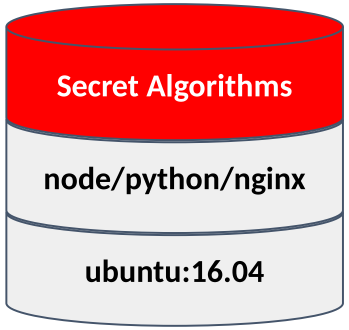
```
{
  "schemaVersion": 2,
  "config": {
 ...
  },
  "layers": [
    {
      "mediaType": "...tar+gzip+enc",
      "size": 32654,
      "digest": "sha256:ee34876d..."
      "annotations": {
        "org…enc.keys": "eefg9...",
      }
    },
    {
      "mediaType": "...tar+gzip+enc",
      "size": 16724,
      "digest": "sha256:ee3a4604..."
      "annotation": {
        "org…enc.keys": "bfc3a...,
                        da8be..."
        }
    },
      ...
  ],
}
```

*+enc = +encrypted abbreviated

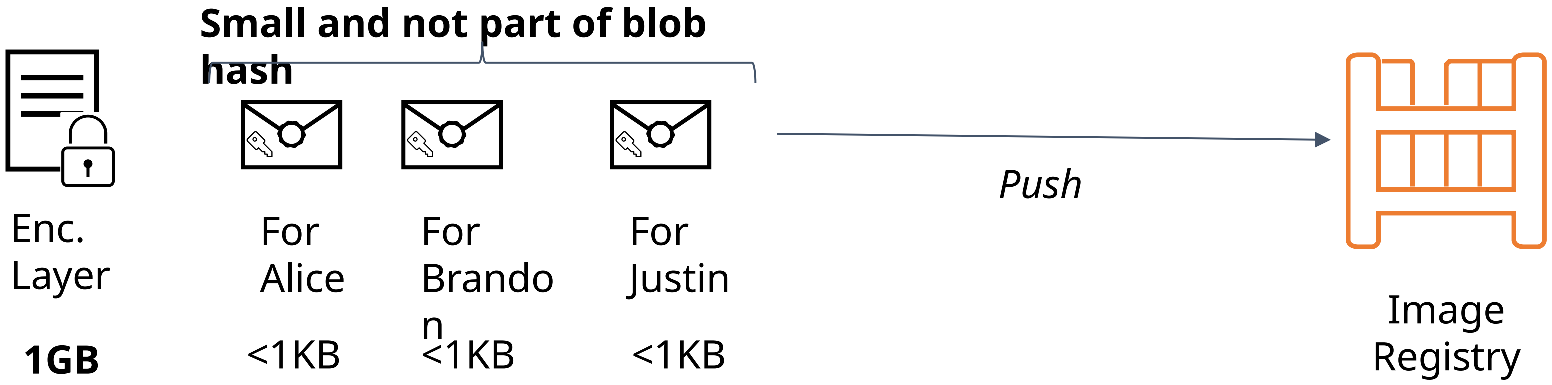
*org…enc.keys annotations represent encryption metadata

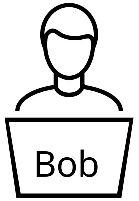
Container Encryption Features

- Encrypt on layers means images can still benefit from deduplication of non-sensitive layers

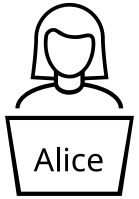


- Encrypt once for multiple recipients. Registry deduplication on large encrypted data blob.





<u>**Bob**</u> *sends an encrypted image to* <u>**Alice**</u>



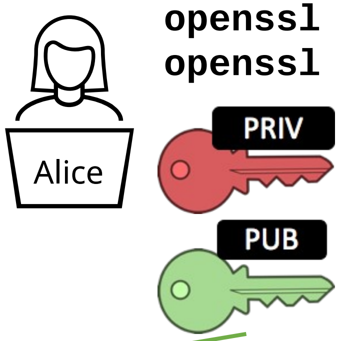
I want to run these images on my Kubernetes cluster!

Bob **Bob** *sends an encrypted image to* **Alice** Alice

Bob *sends an encrypted image to* **Alice**

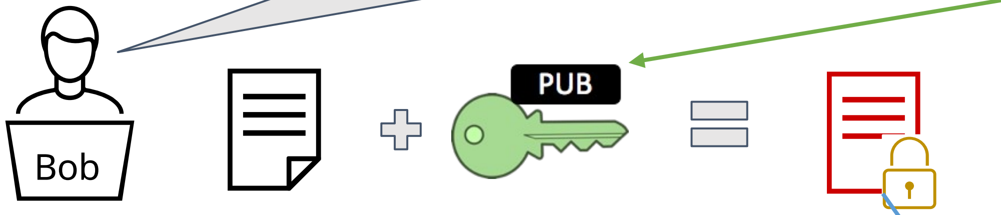
1 Alice creates an RSA private/public keypair, publishing her public key

```
openssl genrsa -out alicePrivate.pem 2048
openssl rsa -in alicePrivate.pem –pubout ...
```

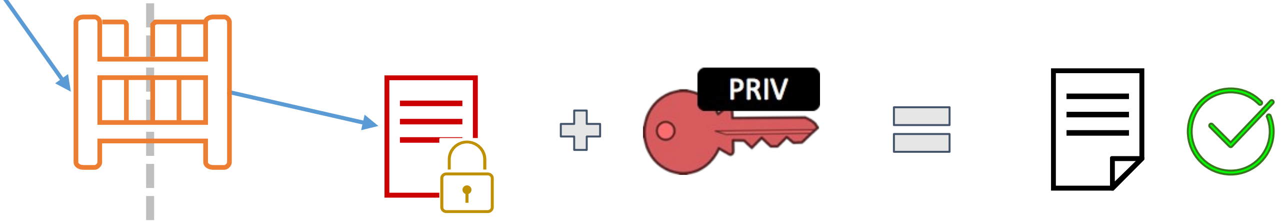


2 Bob encrypts his image and pushes it to the registry

buildah push --encryption- `alicePublic.pem` ...



3 **Alice’s k8s cluster** ~~Alice~~ pulls and decrypts the image with her private key



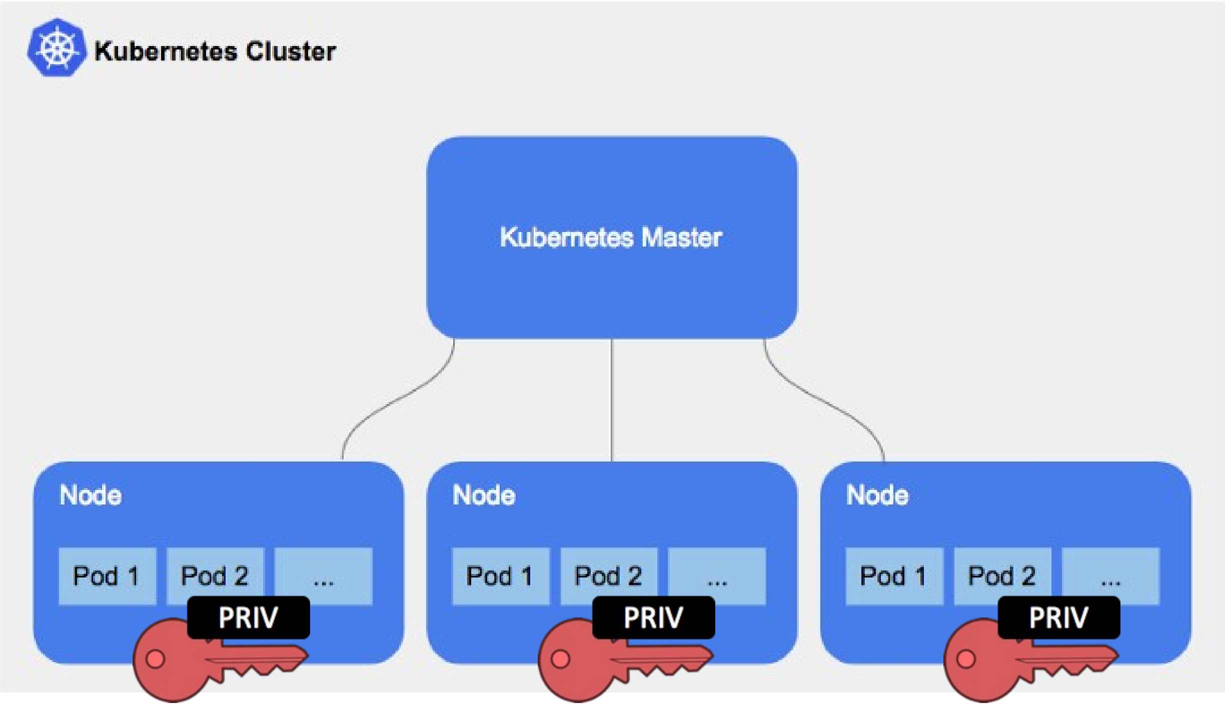
Adding Decryption to K8s Nodes

How is decryption done in k8s?

- Decryption is done when image is pulled by the container runtime, i.e. containerd/cri-o
- Configuration is on the node level

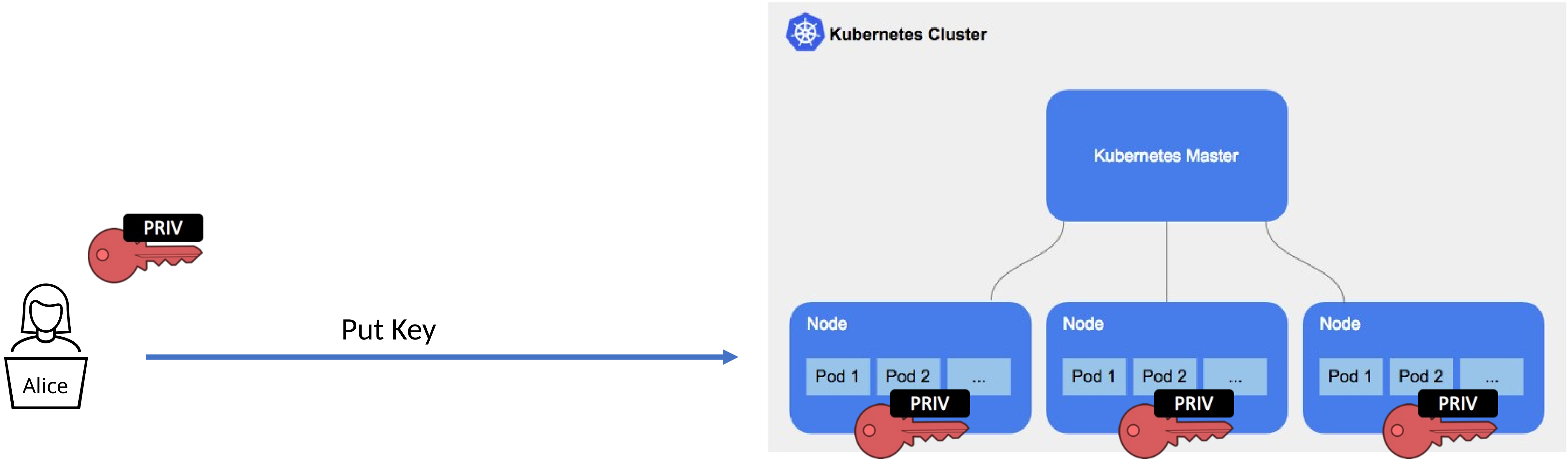
Runtime	Key Location
cri-o	/etc/crio/keys
containerd	<set in config.toml>

- Alice configures the nodes in the Kubernetes cluster with the private key for decryption
- The encrypted image is pulled from the registry and decrypted with the private key provided

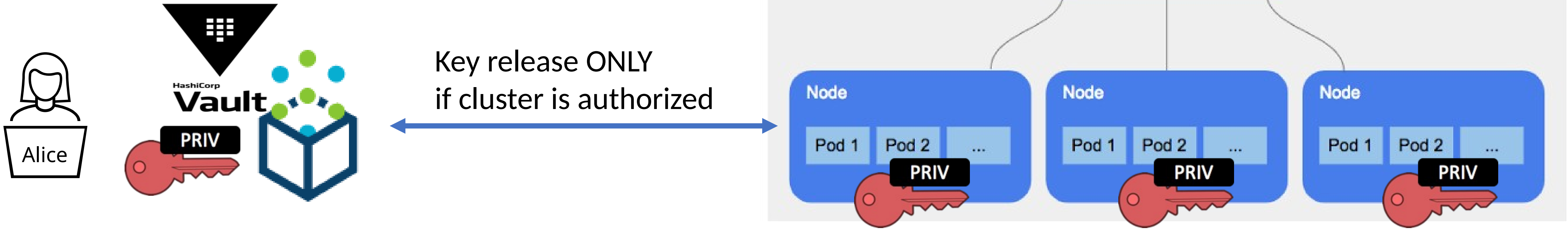


Cluster Decryption Demo

Adding Decryption to K8s Nodes



Adding Decryption to K8s Nodes

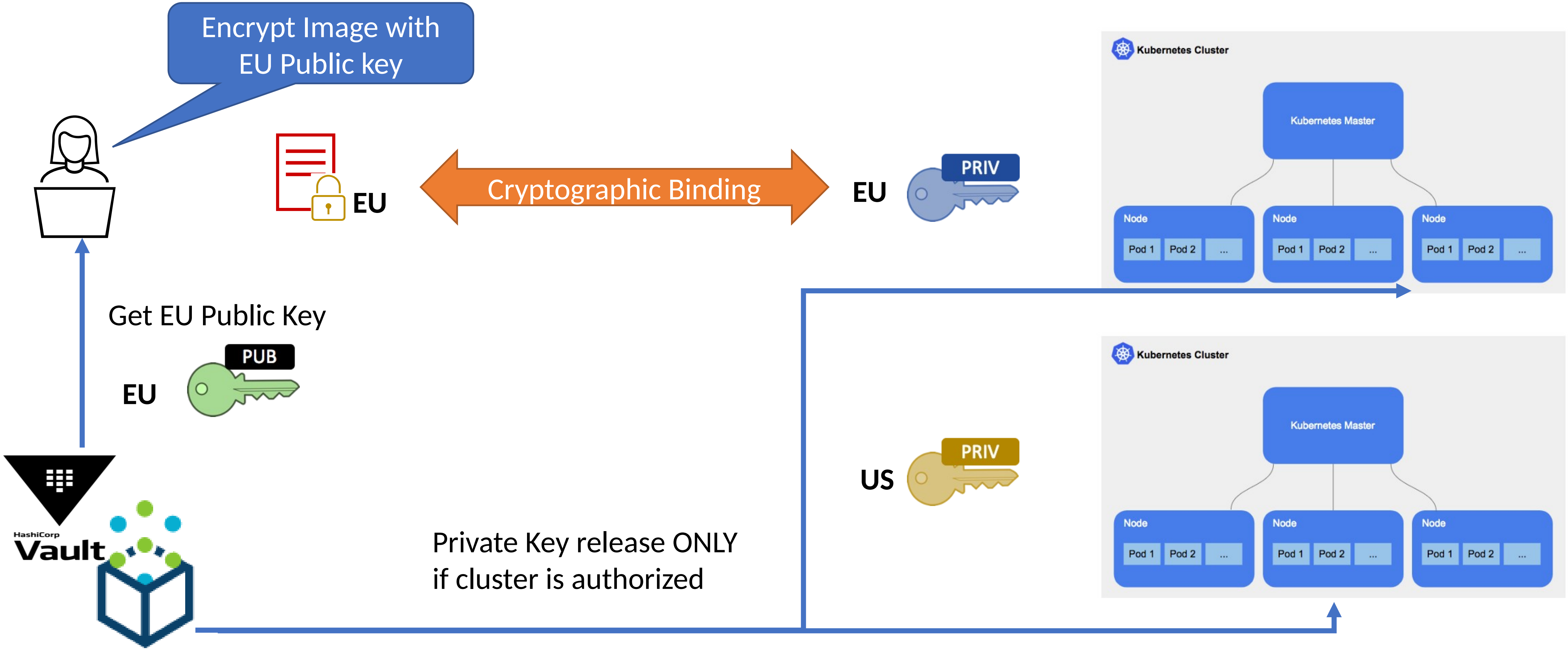


Key release ONLY
if cluster is authorized

Decryption keys can be protected behind cluster authorization

- Proof stemming from cluster CA
- Proof stemming from hardware root of trust attestation

Geofencing Execution



Encrypt Image with EU Public key

Cryptographic Binding

Get EU Public Key

Private Key release ONLY if cluster is authorized

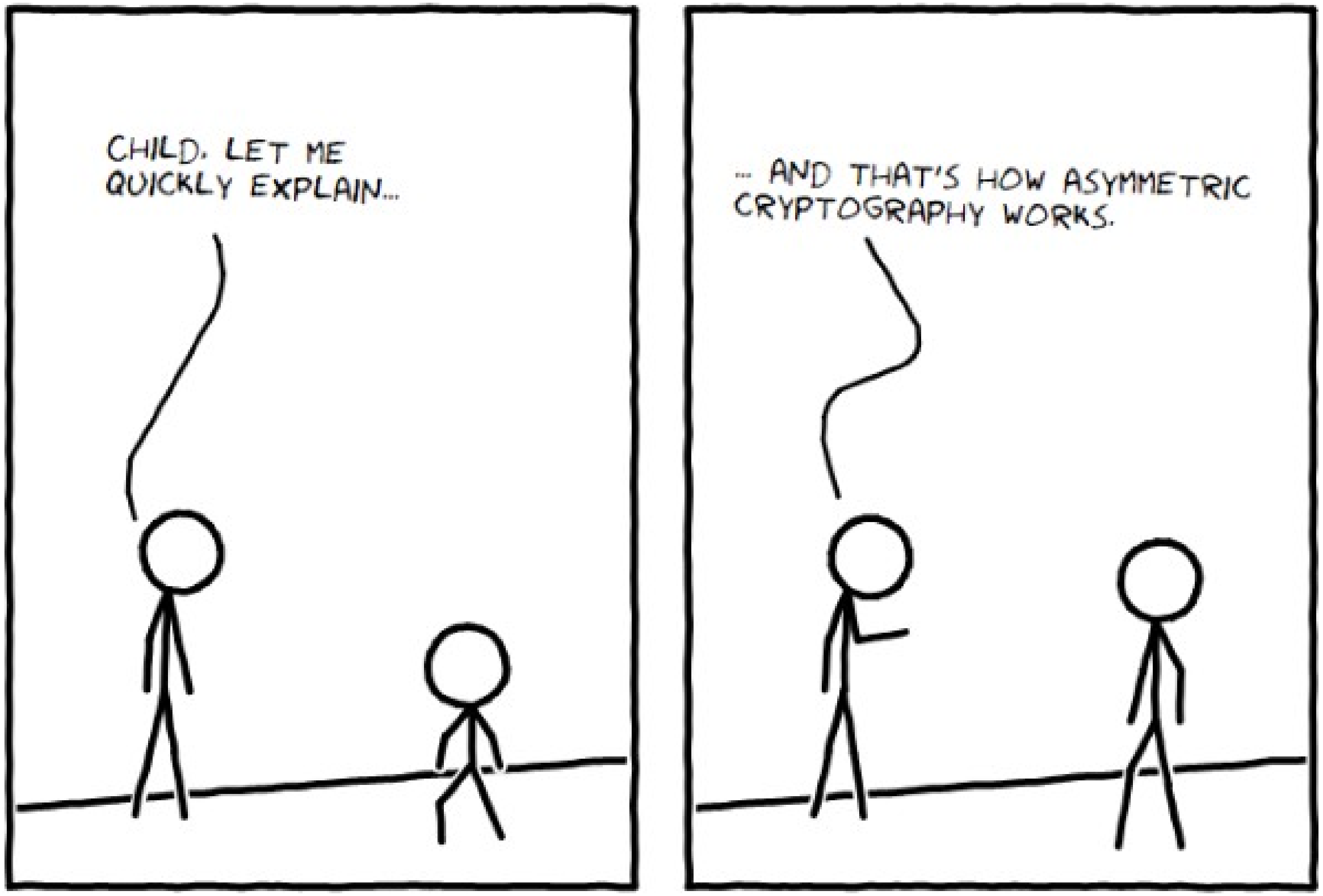
Summary

- Encrypted Container Images can be used for confidentiality of images
- With key management, can create geofencing based policies

- Encrypted Container Images is supported today in:
  - Runtimes: Containerd, Cri-o
  - Build Tools: Buildah, Skopeo
  - Registries: Docker Distribution

- **Call for contribution!**
  - Build Tools: kaniko, docker CLI
  - Registries: Quay

Encryption Primer

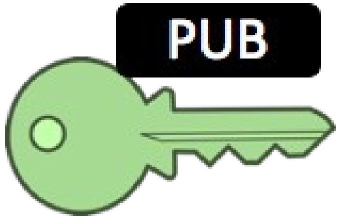

CHILD. LET ME QUICKLY EXPLAIN...
... AND THAT'S HOW ASYMMETRIC CRYPTOGRAPHY WORKS.

https://www.outsystems.com/blog/posts/how-to-teach-child-about-asymmetric-cryptography/

Encryption Primer - Assym Enc.

This is a secret message.

+

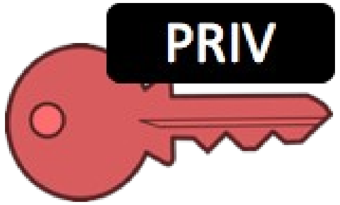


=

903nsvlvn391x
kshu9282jks91
0alfde=

903nsvlvn391x
kshu9282jks91
0alfde=

+



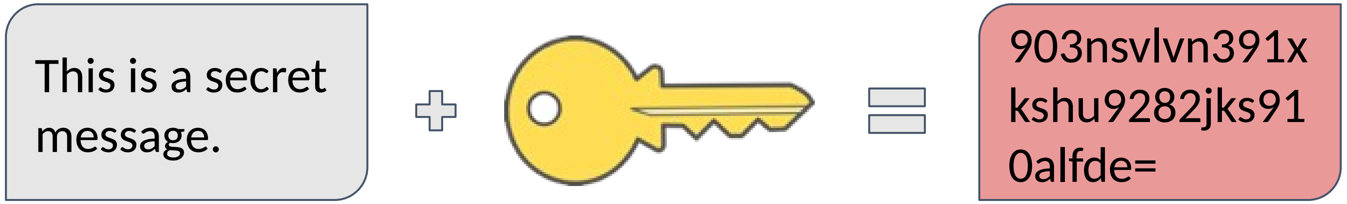
=

This is a secret message.

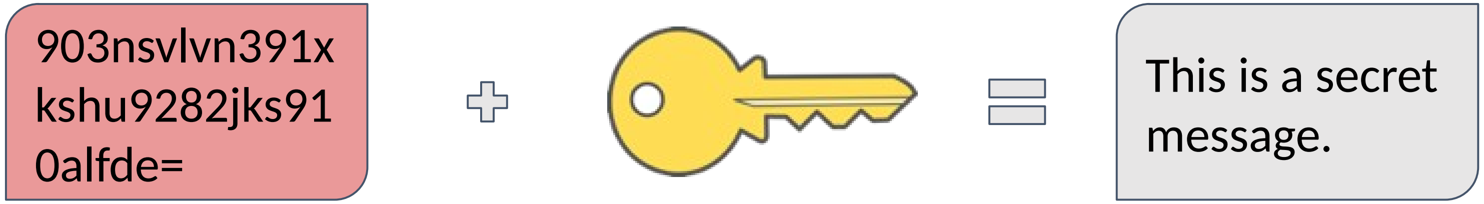
👍 Addresses key sharing: Each user has a Public-Private key pair, where Public Key is not secret, can be published.

👎 Slow

Encryption Primer - Symm Enc.



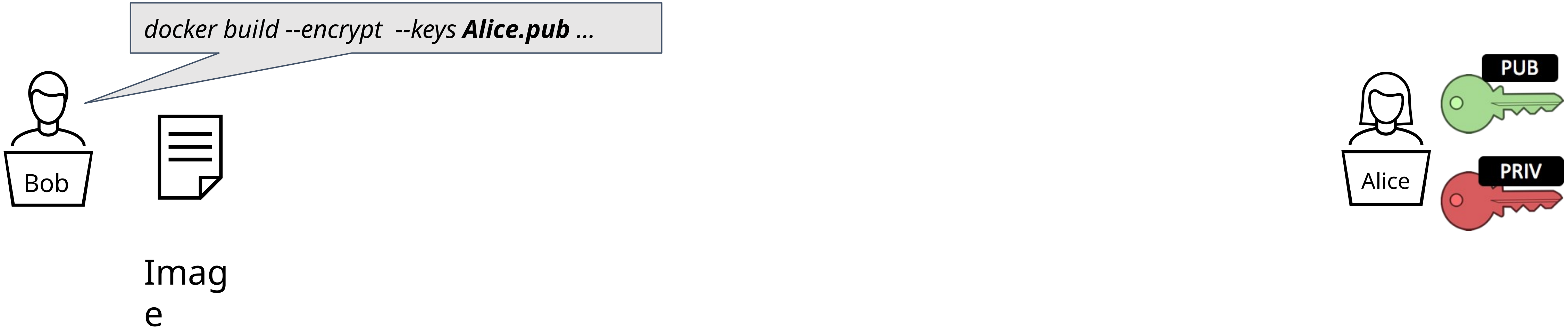
This is a secret message. + = 903nsvlvn391x kshu9282jks91 0alfde=



903nsvlvn391x kshu9282jks91 0alfde= + = This is a secret message.

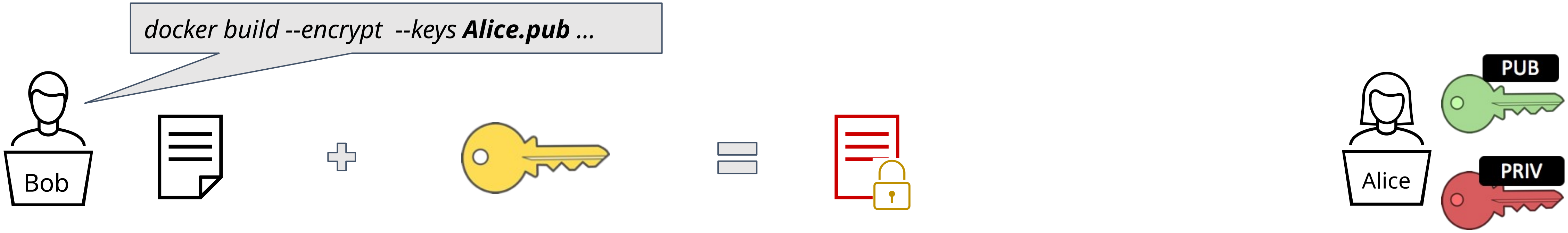
- 👍 Fast, good for large data
- 👎 Key needs to be securely shared somehow

Encrypt and distribution flow



```
docker build --encrypt  --keys Alice.pub ...
```

Encrypt and distribution flow



*docker build --encrypt --keys **Alice.pub** ...*

Bob

Alice

PUB

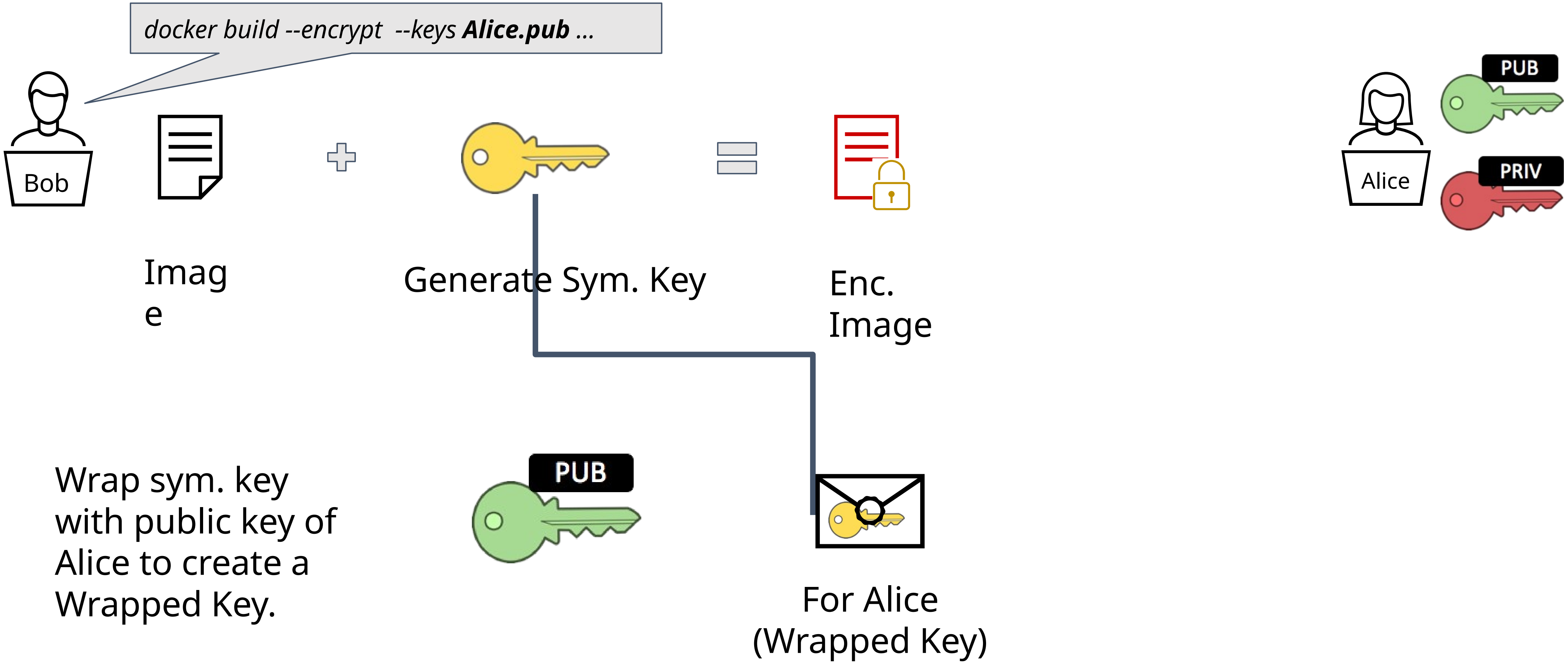
PRIV

Imag
e

Generate Sym. Key

Enc.
Image

Encrypt and distribution flow



```
docker build --encrypt  --keys Alice.pub ...
```

Image

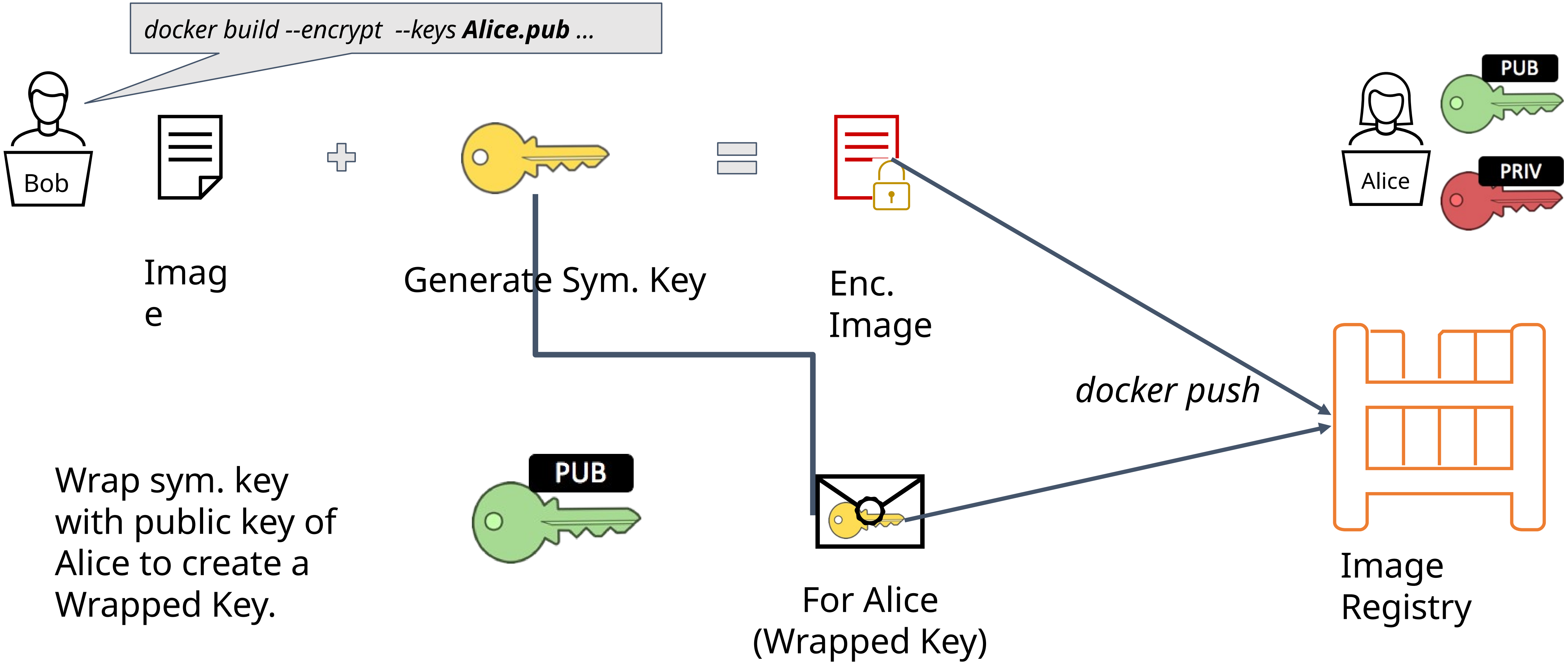
Generate Sym. Key

Enc. Image

Wrap sym. key with public key of Alice to create a Wrapped Key.

For Alice (Wrapped Key)

Encrypt and distribution flow



*docker build --encrypt --keys **Alice.pub** ...*

Imag
e

Generate Sym. Key

Enc.
Image

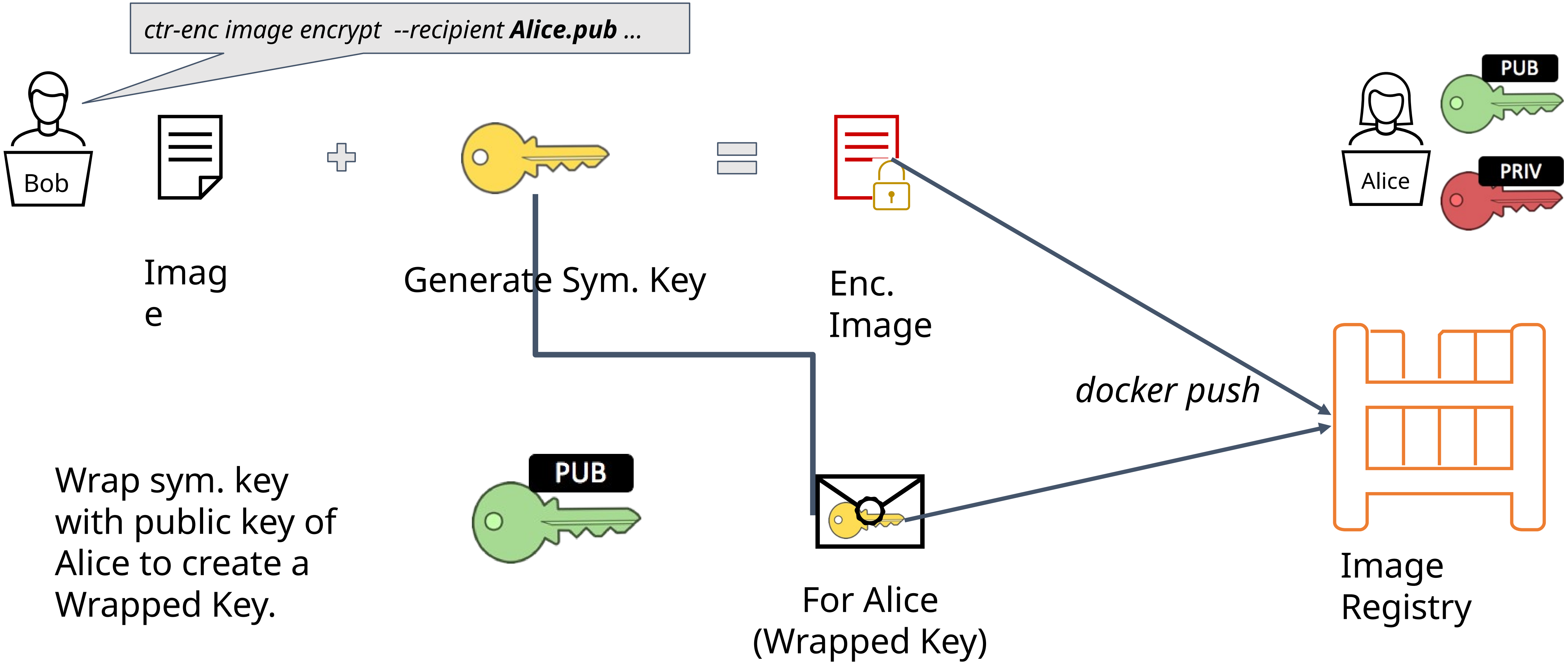
docker push

Image
Registry

Wrap sym. key
with public key of
Alice to create a
Wrapped Key.

For Alice
(Wrapped Key)

Encrypt and distribution flow



```
ctr-enc image encrypt  --recipient Alice.pub ...
```

Image

Generate Sym. Key

Enc. Image

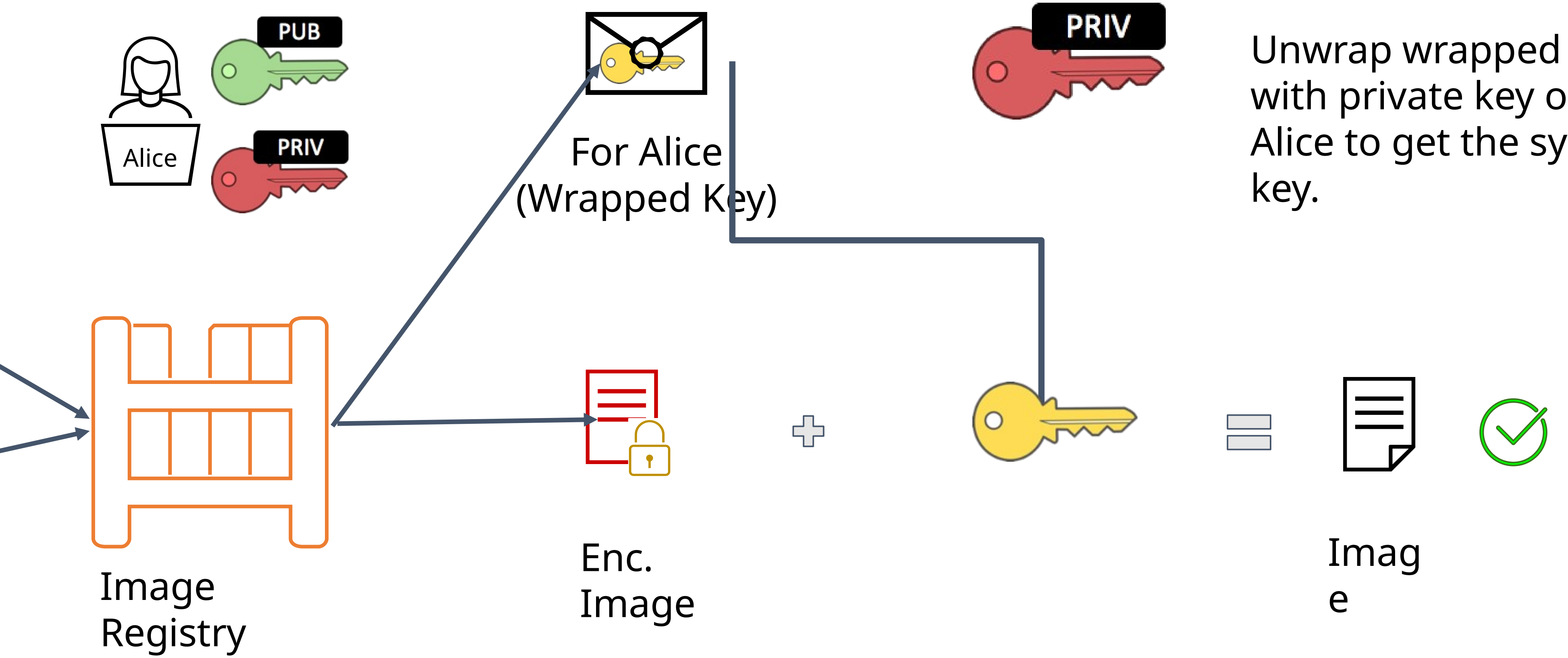
docker push

Image Registry

Wrap sym. key with public key of Alice to create a Wrapped Key.

For Alice (Wrapped Key)

Encrypt and distribution flow



Unwrap wrapped key with private key of Alice to get the sym. key.