Deploying a Secure Docker Registry

Nigel Brown

@n_brownuk www.windsock.io

Module Outline

Interacting with an insecure registry

Securing communication with a registry

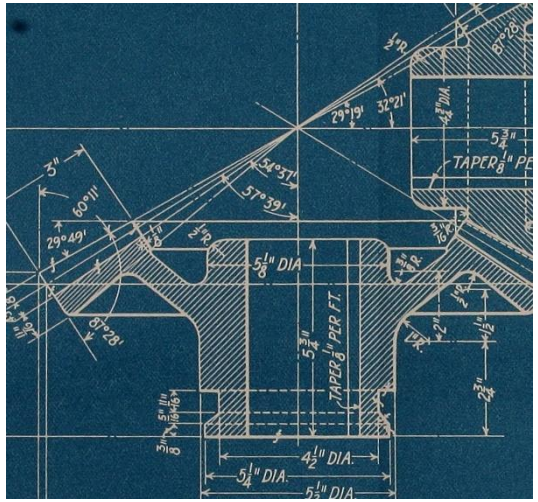
Access control methods for registries

Configuring authentication techniques

What Is a Docker Registry?

Storage and delivery mechanism

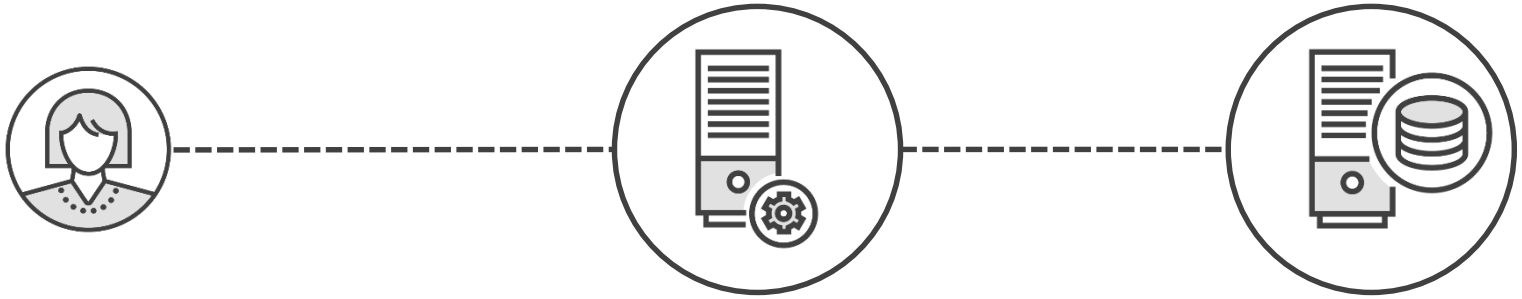
Serves registry HTTP API v2



Open source distribution project

Many commercial implementations

Addressing a Docker Registry




```
$ docker image pull registry.acme.com:5000/app:1.0
```

```
$ docker image pull registry.acme.com:5000/app:1.0
Error response from daemon: Get
https://registry.acme.com:5000/v2/: http: server gave HTTP
response to HTTPS client
```

Communicating with an Insecure Registry

Docker daemon expects to communicate with registry using TLS

```
# Config flag for daemon to use insecure registry
--insecure-registry=“registry.acme.com:5000”

# JSON key/value used in daemon config file
{
    “insecure-registries”: [“registry.acme.com:5000”]
}
```

Daemon Configuration for Insecure Registries

When defined in daemon config file, send `SIGHUP` to daemon

A registry exposed on 127.0.0.1/8 network, is deemed insecure

Securing a Docker Registry

The Docker daemon is the client, the Docker registry is the server

Daemon authenticates registry, or both parties authenticate the other

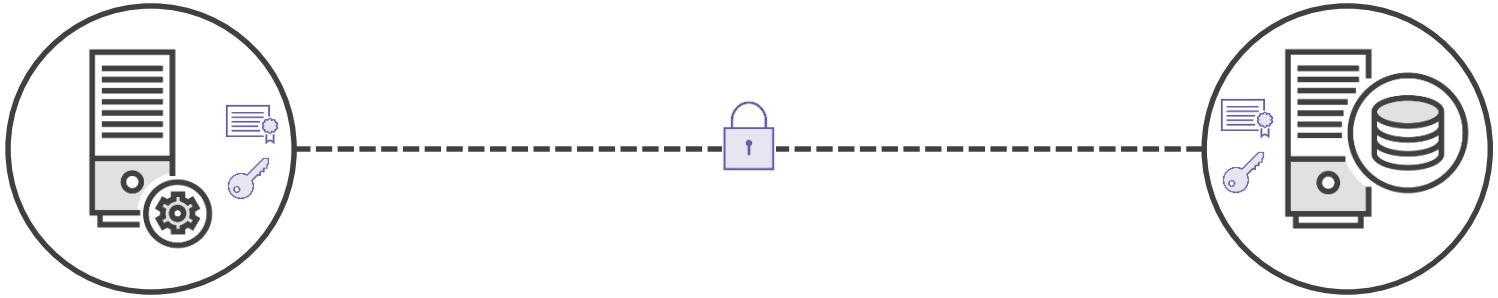
TLS configuration of registry can be automated with Let's Encrypt

```
version: 0.1
log:
  fields:
    service: registry

<snip>

http:
  addr: 0.0.0.0:5000
  tls:
    certificate: /certs/cert.pem
    key: /certs/key.pem

<snip>

health:
  storagedriver:
    enabled: true
    interval: 10s
    threshold: 3
```

- Registry configured using YAML-based config file

- TLS is defined under the `http` primary key
- The `tls` sub-key is entirely optional
- Keys can be overridden with variables, e.g. `REGISTRY_HTTP_TLS_CERTIFICATE`

Configuring Mutual TLS



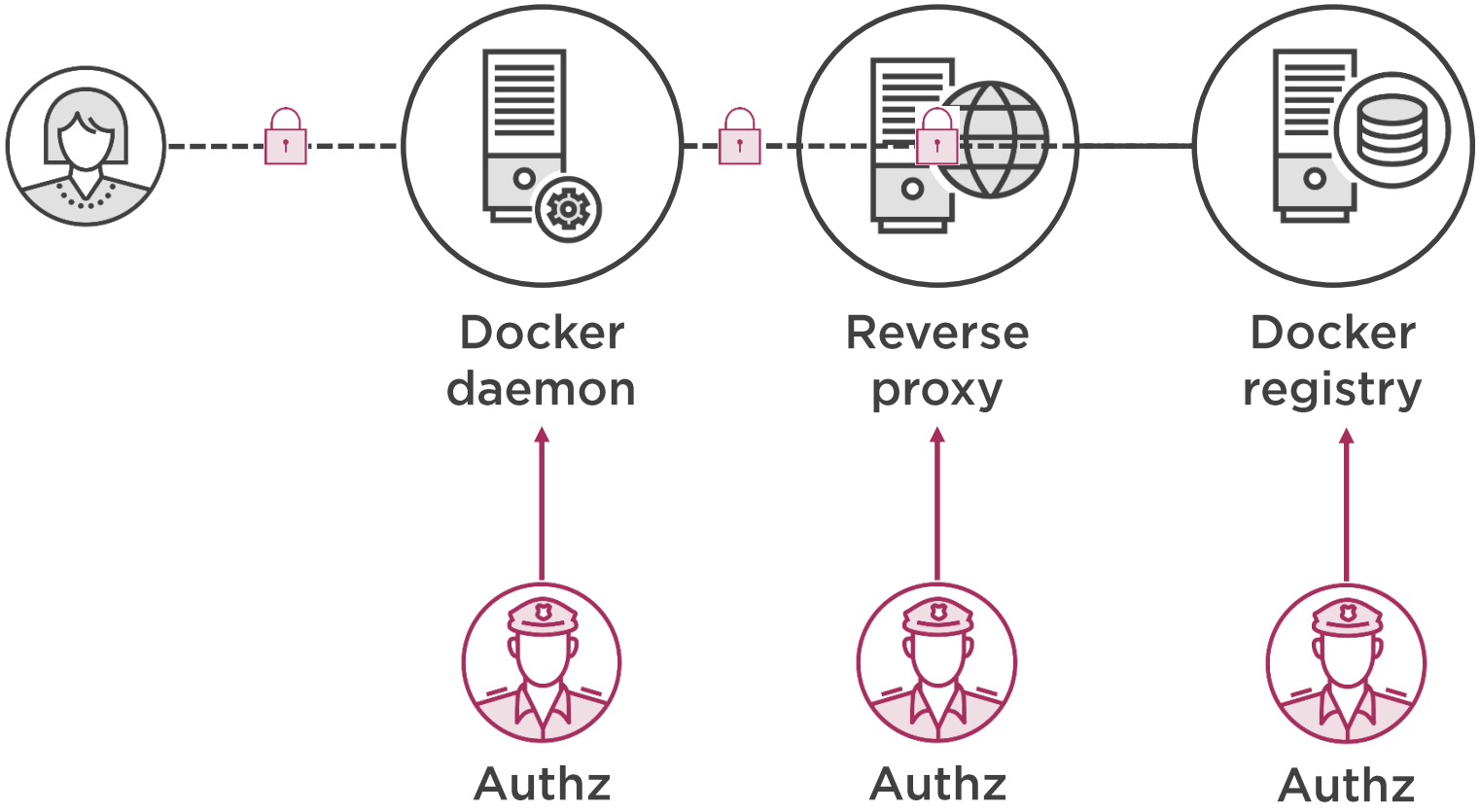
Docker daemon

Docker registry

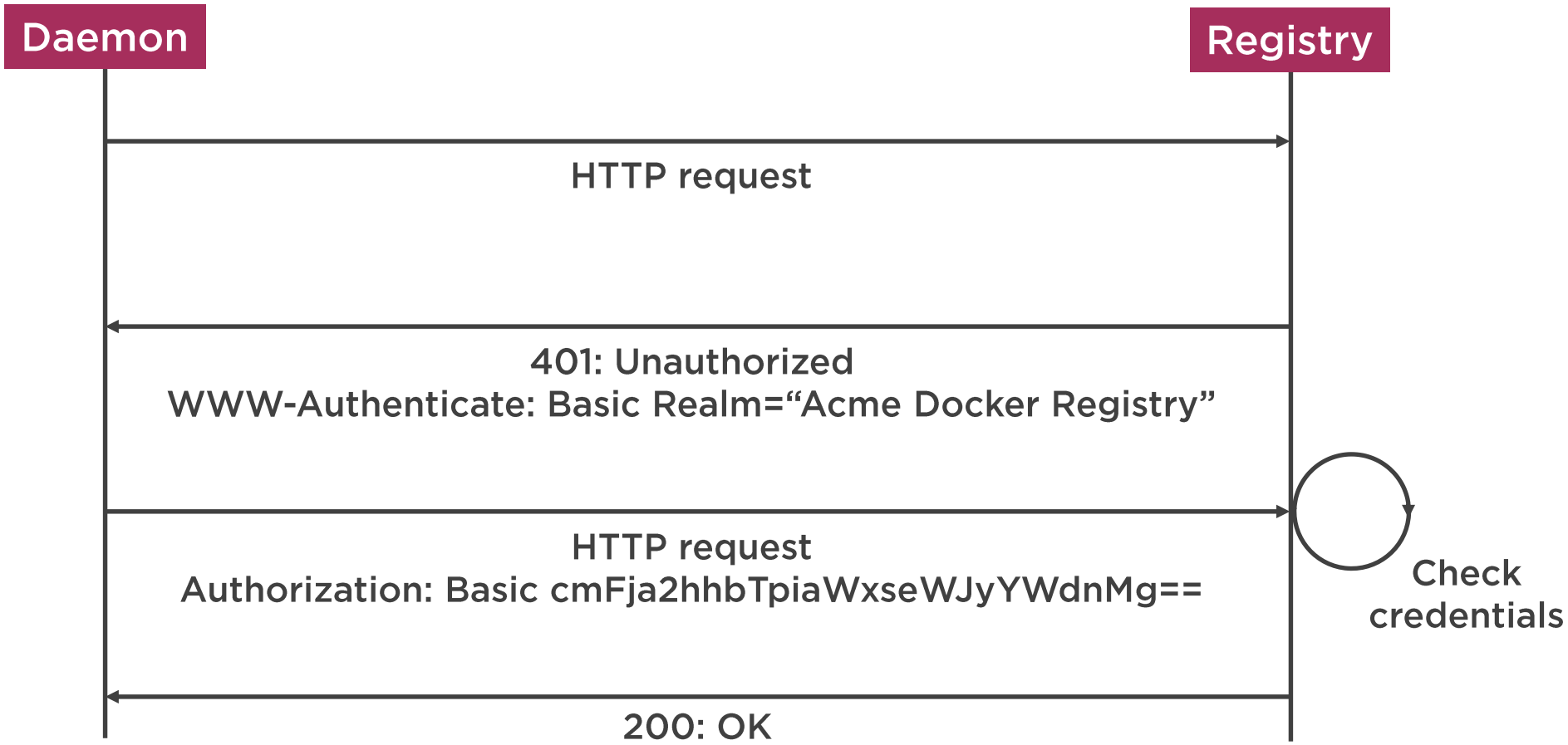
```
/etc/docker/certs.d/calculus.lan:443/
├── ca.crt
├── wolff.cert
└── wolff.key
```

```
{
  <snip>
  http:
    tls:
      certificate: /certs/cert.pem
      key: /certs/key.pem
  <snip>clientcas:
}       - /certs/ca.pem
```

Authorization Options



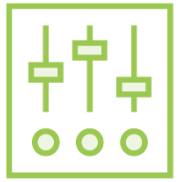
Basic Access Authentication



Basic Auth Characteristics

The registry must be configured with TLS for safe use of basic auth

The daemon limits registry operations to pushing and pulling images



No fine-grain access control, accessing the registry is all or nothing

```
auth:
  htpasswd:
    realm: calculus.lan:443
    path: /auth/htpasswd
```

Configuring the Registry for Basic Auth

The primary key for configuring authentication is `auth`

Basic auth is configured with the `htpasswd` **sub-key**

```
$ docker login -u rackham calculus.lan:443
Password:

Login Succeeded
```

Logging into a Docker Registry

Use the `docker login` command with the registry name

Use a 'credential helper' to store your registry credentials

- https://dockr.ly/2wYCAES

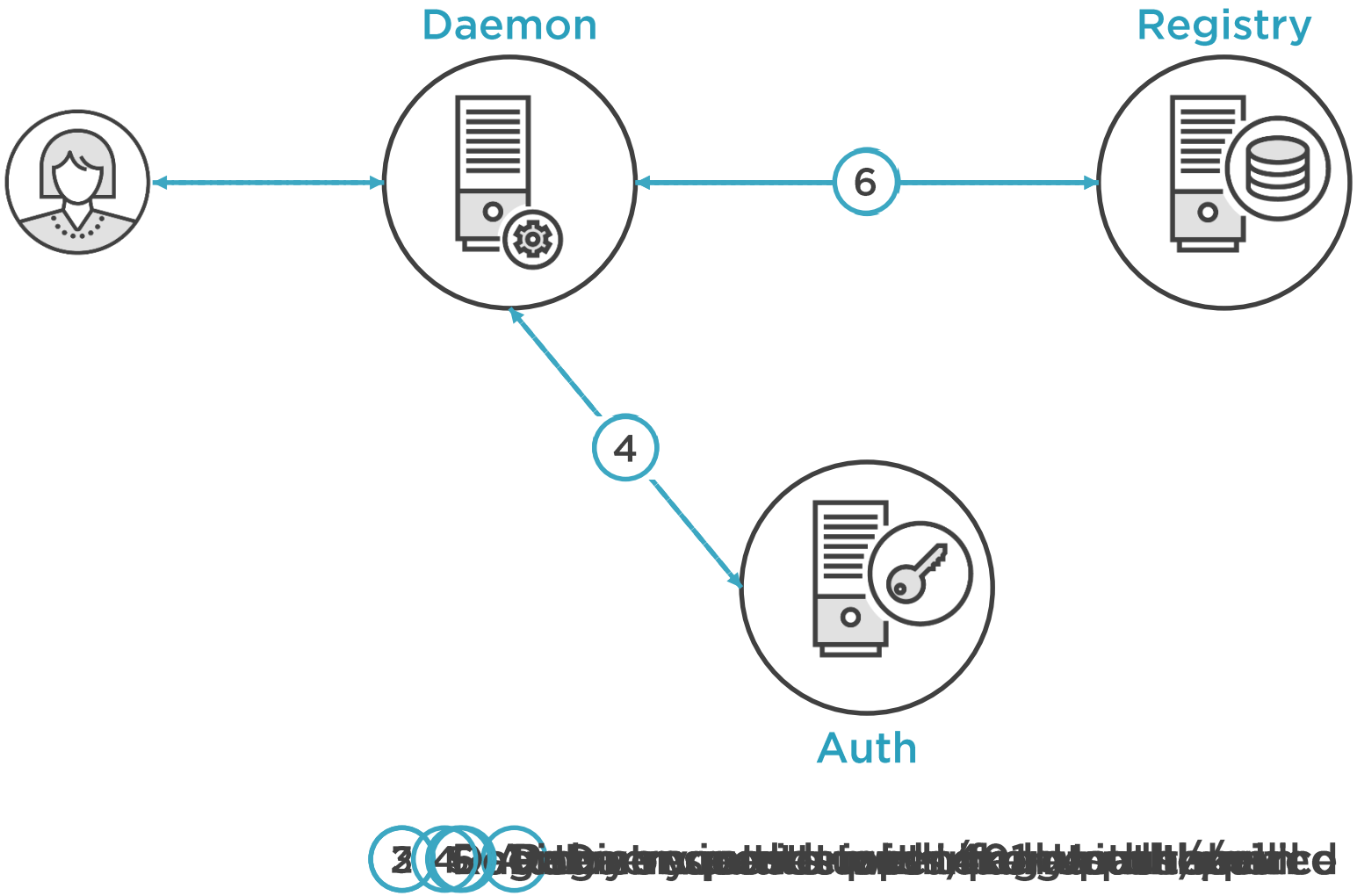
Advanced Registry Authentication

The Docker registry supports token-based authentication

The Distribution project provides a spec, but not an implementation

Examples include Keycloak and docker_auth (https://git.io/vhmLb)

Token-based Authentication



```
auth:
 token:
   realm: https://auth.lan:5001/auth
   service: Acme Docker Registry
   issuer: Acme Auth Server
   rootcertbundle: /certs/auth.pem
```

Configuring the Registry for Token-based Auth

Token-based auth is configured with the `token` sub-key

Further keys define the realm, service, and token issuer

If required, the `rootcertbundle` key points to a certificate

docker_auth

Authentication

Static list, Google/GitHub auth, LDAP bind, or MongoDB User Collection

Authorization

Static ACL, MongoDB-backed ACL, external program

Module Summary

Docker daemons expect secure communication with Docker registries

Mutual TLS provides the most secure configuration

Access control can be implemented using basic or token-based authentication