Security

HP

Security features

- Access Control (Authentication, Authorisation)
- SASL Authentication
- TLS Support

Access Control

- authentication as "identifying who the user is“
- authorisation as "determining what the user is and isn't allowed to do."

HP

Default Virtual Host and User

- a virtual host : /
- guest with a default password of guest
  - full access to the / virtual host.
- advisable to **delete** the guest user or **change the password**

HP

"guest" user

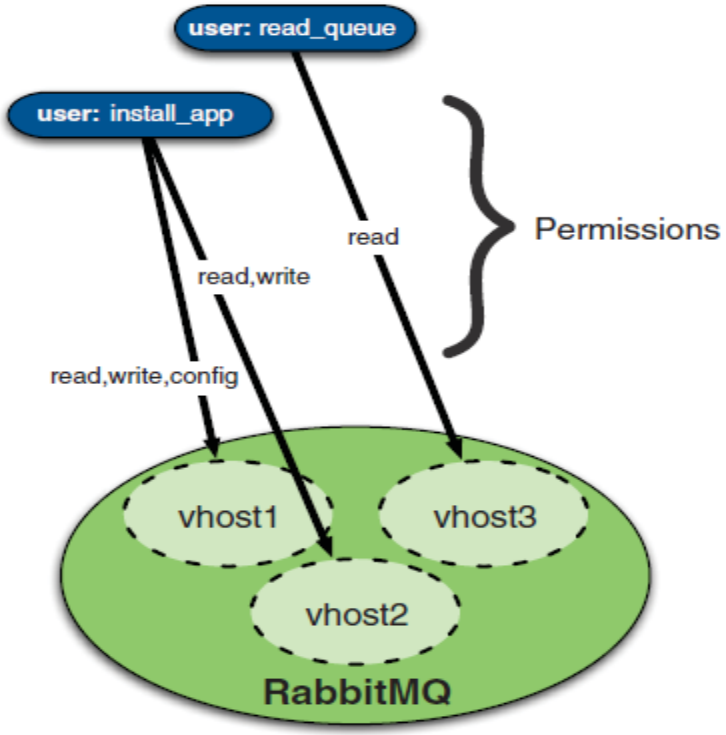
- Can only connect via localhost
- Configured via the loopback_users item in the configuration file

```
loopback_users = none
```

allow the guest user to connect from a remote host

Permissions Working

- RabbitMQ client needs to specifies a virtual host while connecting to a server.
- The server checks the permissions to access the virtual hosts



HP

Permissions Working

- Resources:
  - i.e. exchanges and queues → named entities inside a particular virtual host
  - denotes a different resource in each virtual host.
- A second level of access control is enforced when certain operations are performed on resources.
- Operations :
  - configure, write and read

HP

Permissions Working

- Configure
  - create or destroy resources, or alter their behaviour
- Write
  - inject messages into a resource
- Read
  - retrieve messages from a resource

- user must have been granted appropriate permissions on resources.

HP

Permissions

- expressed as a triple of regular expressions - one each for configure, write and read - on per-vhost basis.
- '^$', i.e. matching nothing
  - covers all resources
  - effectively stops user from performing any operation.
- Standard AMQP resource names are prefixed with amq.
  - '^(amq\.gen.*|amq\.default)$' gives a user access to server-generated names and the default exchange.

Permissions

- '' is a synonym for '^$' and restricts permissions in the exact same way.
- cache the results of access control checks on a per-connection or per-channel basis.

HP

AMQP commands permission checks

AMQP 0-9-1 Operation		configure	write	read
exchange.declare	(passive=false)	exchange		
exchange.declare	(passive=true)			
exchange.declare	(with AE)	exchange	exchange (AE)	exchange
exchange.delete		exchange		
queue.declare	(passive=false)	queue		
queue.declare	(passive=true)			
queue.declare	(with DLX)	queue	exchange (DLX)	queue
queue.delete		queue		

Passive : If set, the server will reply with Declare-Ok if the queue already exists with the same name, and raise an error if not

Alternate Exchange : an exchange use when unable to route (i.e. either because there were no bound queues our no matching bindings

Dead Letter Exchange :Messages from a queue can be 'dead-lettered'; that is, republished to another exchange when any message is rejected

HP

AMQP commands permission checks

AMQP 0-9-1 Operation	configure	write	read
exchange.bind		exchange (destination)	exchange (source)
exchange.unbind		exchange (destination)	exchange (source)
queue.bind		queue	exchange
queue.unbind		queue	exchange
basic.publish		exchange	
basic.get			queue
basic.consume			queue
queue.purge			queue

HP

A & A Backends

- Authentication and authorisation are pluggable.
- Plugins can provide implementations of
  - authentication ("authn") backends
  - authorisation ("authz") backends
- It is possible for a plugin to provide both.
  - E.x the internal, **LDAP** and **HTTP** backends do so.

HP

Combining Backends

- possible to use multiple backends for authn or authz
- use rabbit.auth_backends configuration key
- the first positive result returned by a backend in the chain is considered to be final

```
# try LDAP first
 auth_backends.1 = ldap
 # fall back to the internal database
auth_backends.2 = internal
```

HP

SASL

- RabbitMQ has pluggable support for various SASL authentication mechanisms.

Mechanism	Description
PLAIN	SASL PLAIN authentication. This is enabled by default in the RabbitMQ server and clients, and is the default for most other clients.
AMQPLAIN	Non-standard version of PLAIN retained for backwards compatibility. This is enabled by default in the RabbitMQ server.
EXTERNAL	Authentication happens using an out-of-band mechanism such as x509 certificate peer verification, client IP address range, or similar. Such mechanisms are usually provided by RabbitMQ plugins.
RABBIT-CR-DEMO	Non-standard mechanism which demonstrates challenge-response authentication. This mechanism has security equivalent to PLAIN, and is **not** enabled by default in the RabbitMQ server.

auth_mechanisms.1 = PLAIN
auth_mechanisms.2 = AMQPLAIN

TLS & SSL Support

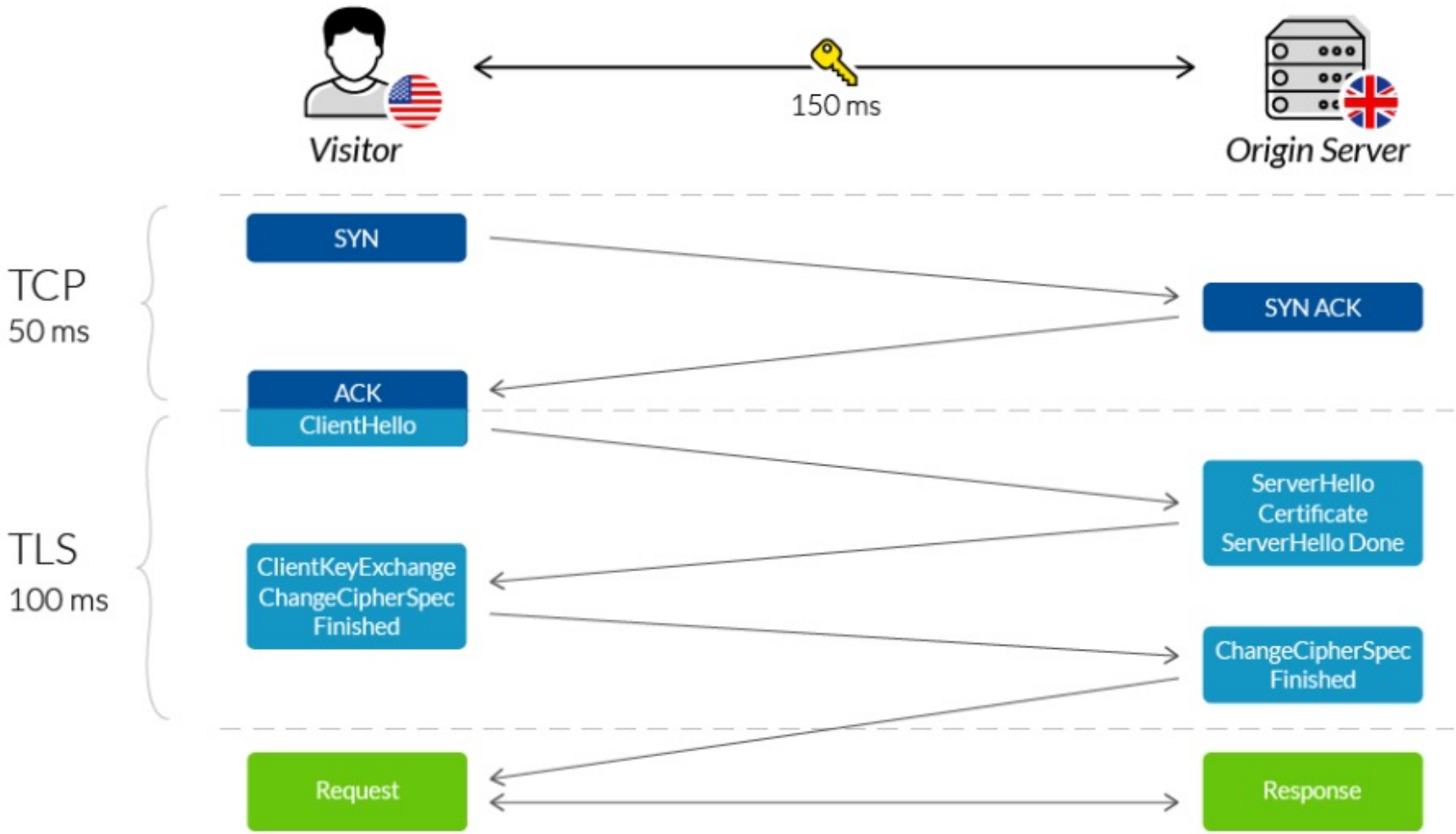
HP

Overview

- inbuilt support for TLS.
  - includes client connections and popular plugins, where applicable, such as **Federation links**.
- can use TLS to encrypt inter-node connections in clusters.

TLS

- What is TLS?



TLS

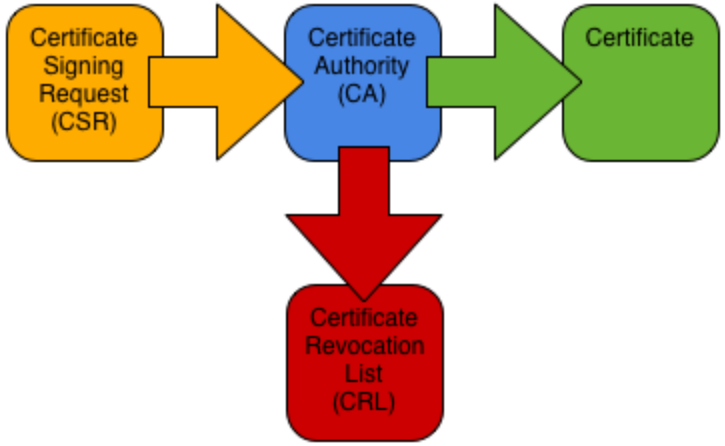
- The Erlang crypto, asn1, public_key, and ssl libraries (applications) must be installed and functional.

- needs TLS and crypto-related modules to be available in the Erlang/OTP installation.

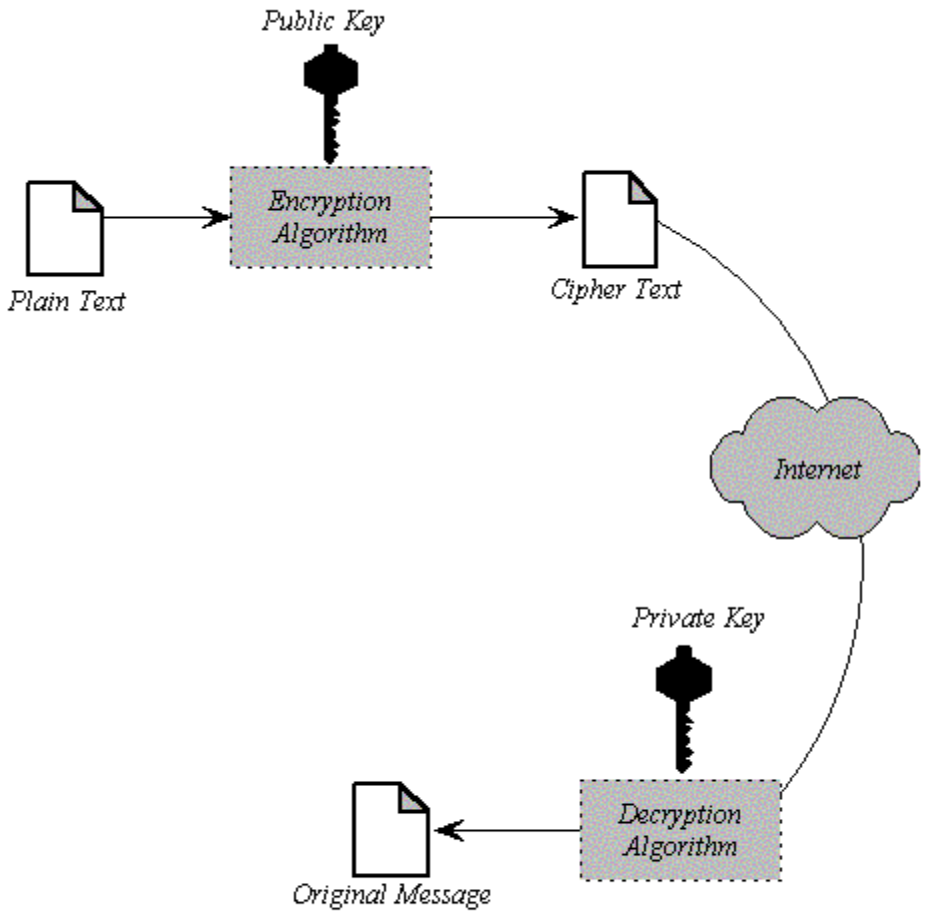
Configuring SSL

- Install OpenSSL
- Create *Certificate Authority*
- Generate signed certificates for the server and clients
- Enabling SSL Support in RabbitMQ

PKI

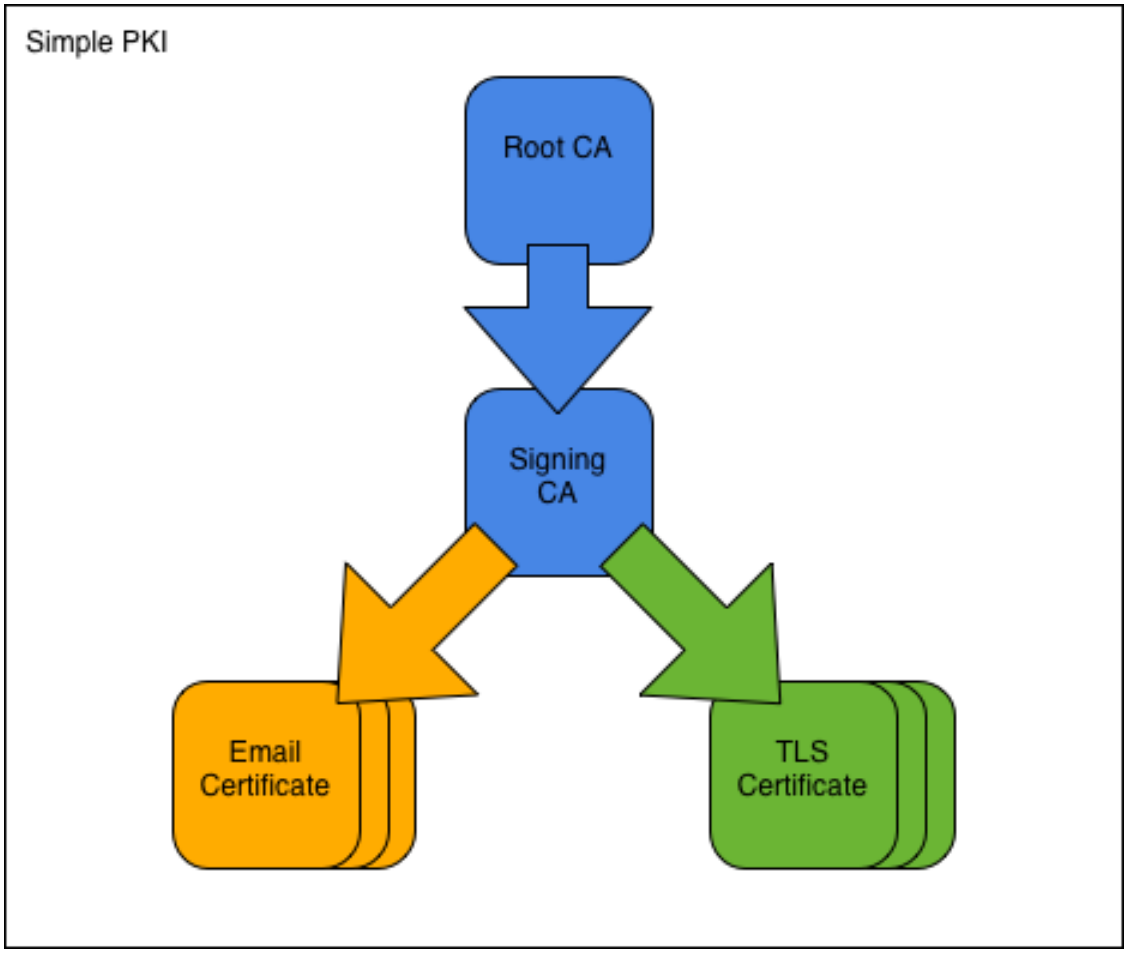


Asymmetric key (public key) encryption



Pubic Key Cryptography

Overview - PKI



Configuring SSL - openssl.cnf

```
[ testca ]
dir = .
certificate = $dir/cacert.pem
database = $dir/index.txt
new_certs_dir = $dir/certs
private_key = $dir/private/cakey.pem
serial = $dir/serial
```

```
[ root_ca_distinguished_name ]
commonName = hostname

[ root_ca_extensions ]
basicConstraints = CA:true
keyUsage = keyCertSign, cRLSign
```

Configuring SSL - *Certificate Authority*

- generate the key and certificates - *Certificate Authority*

```
#openssl req -x509 -config openssl.cnf -newkey rsa:2048
-days 365 -out cacert.pem -outform PEM /
 -subj /CN=MyTestCA/  -nodes
#openssl x509 -in cacert.pem -out cacert.cer -outform
DER
```

The process for creating server and client certificates is very similar. – Next Slide
The only difference is the *key Usage* field that is added when signing the certificate

Server Certificate

```
# openssl genrsa -out key.pem 2048
# openssl req -new -key key.pem -out req.pem -outform PEM
-subj /CN=$(hostname)/O=server/ -nodes
# cd ../testca
# openssl ca -config openssl.cnf -in ../server/req.pem –out ../server/cert.pem -notext -batch -
extensions server_ca_extensions
# cd ../server
# openssl pkcs12 -export -out keycert.p12 -in cert.pem -
inkey key.pem -passout pass:MySecretPassword
```

private key

generate the CSR

PEM → PKCS12

Sign the certificate with the CA private key using the CSR

Enabling SSL Support in RabbitMQ

- provide to RabbitMQ the location of the *root* certificate, the server's certificate file, and the server's key
- Specify the socket

```
-rabbit ssl_listeners
```

{"127.0.0.1", 5671}

```
[
  {rabbit, [
     {ssl_listeners, [5671]},
     {ssl_options, [{cacertfile,"/path/to/testca/cacert.pem"},
                    {certfile,"/path/to/server/cert.pem"},
                    {keyfile,"/path/to/server/key.pem"},
                    {verify,verify_peer},
                    {fail_if_no_peer_cert,false}]}
   ]}
].
```

rabbit.log

- shows that RabbitMQ server is up and running and listening for ssl connections.

```
=INFO REPORT==== 9-Aug-2010::15:10:55 ===
started TCP Listener on 0.0.0.0:5672

=INFO REPORT==== 9-Aug-2010::15:10:55 ===
started SSL Listener on 0.0.0.0:5671
```

Managing users

Managing users

```
$ ./rabbitmqctl add_user cashing-tier cashMe1
Creating user "cashing-tier" ...
...done.
```

```
$ ./rabbitmqctl list_users
Listing users ...
cashing-tier
guest
...done.
```

```
$ ./rabbitmqctl delete_user cashing-tier
Deleting user "cashing-tier" ...
...done.
```

```
$ ./rabbitmqctl change_password cashing-tier compl3xPassword
Changing password for user "cashing-tier" ...
...done.
```

Managing users

```
$ ./rabbitmqctl set_permissions -p oak \
-s all cashing-tier "" "checks-.*" ".*"
Setting permissions for user "cashing-tier" in vhost "oak" ...
...done.
```

```
$ ./rabbitmqctl list_permissions -p oak
Listing permissions in vhost "oak" ...
cashing-tier            checks-.*       .*      all
...done.
```

Managing users

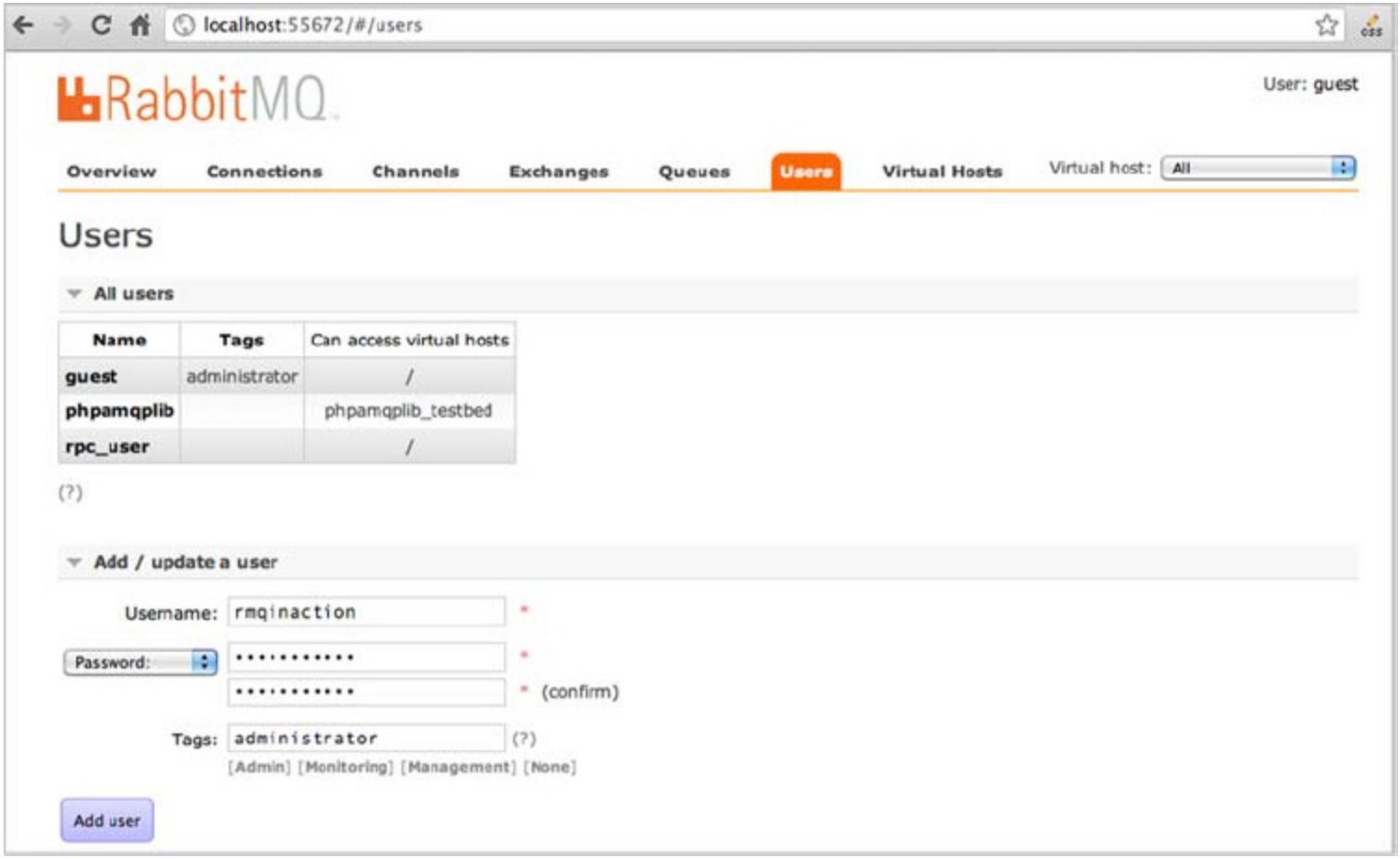
```
$ ./rabbitmqctl clear_permissions -p oak cashing-tier
Clearing permissions for user "cashing-tier" in vhost "oak" ...
...done.
```

```
$ ./rabbitmqctl list_permissions -p oak
Listing permissions in vhost "oak" ...
...done.
```

```
$ ./rabbitmqctl list_user_permissions cashing-tier
Listing permissions for user "cashing-tier" ...
oak             checks-.*       .*      all
sycamore        .*      .*      .*      all
...done.
```

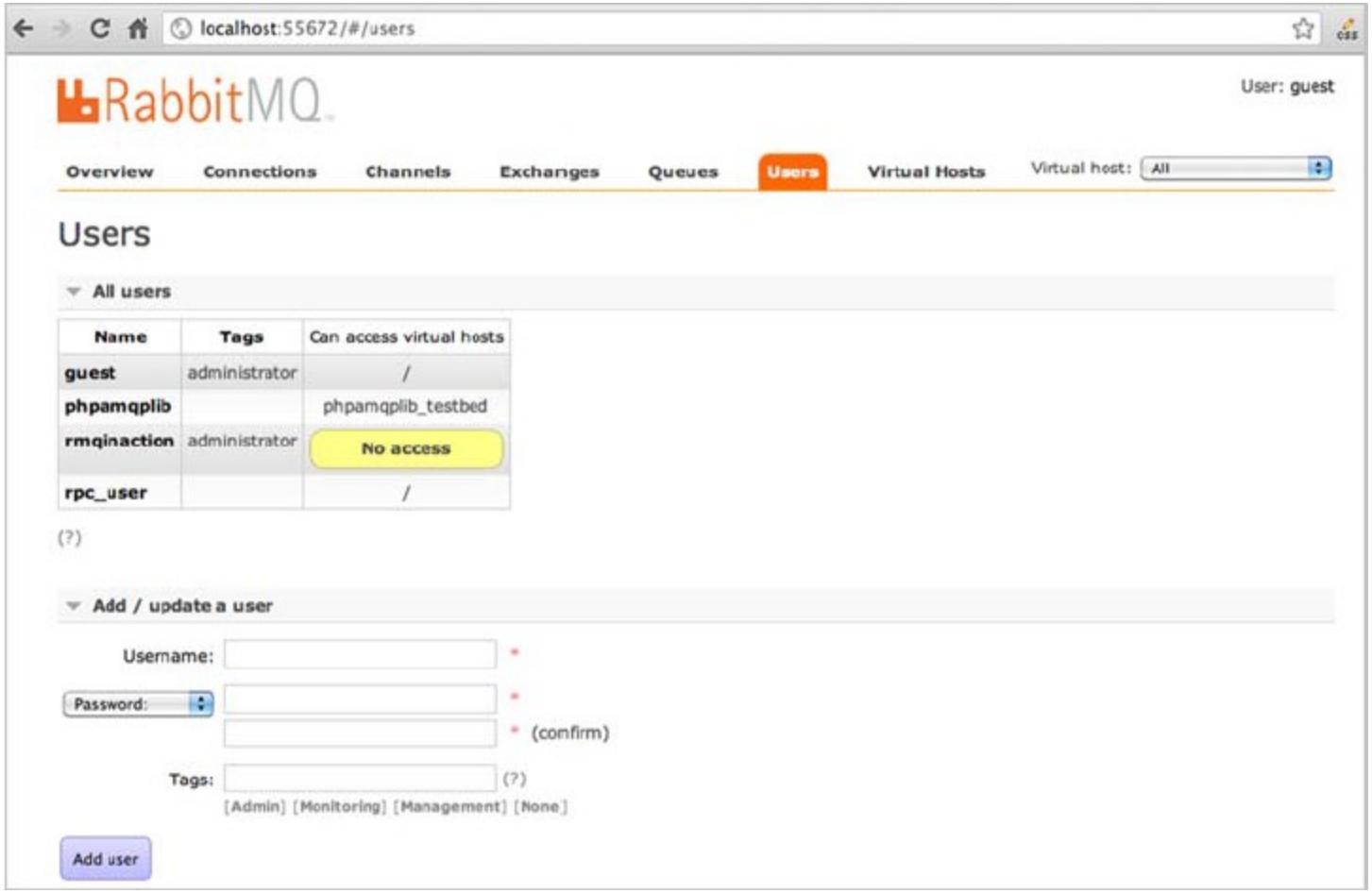
Managing users

- Add users



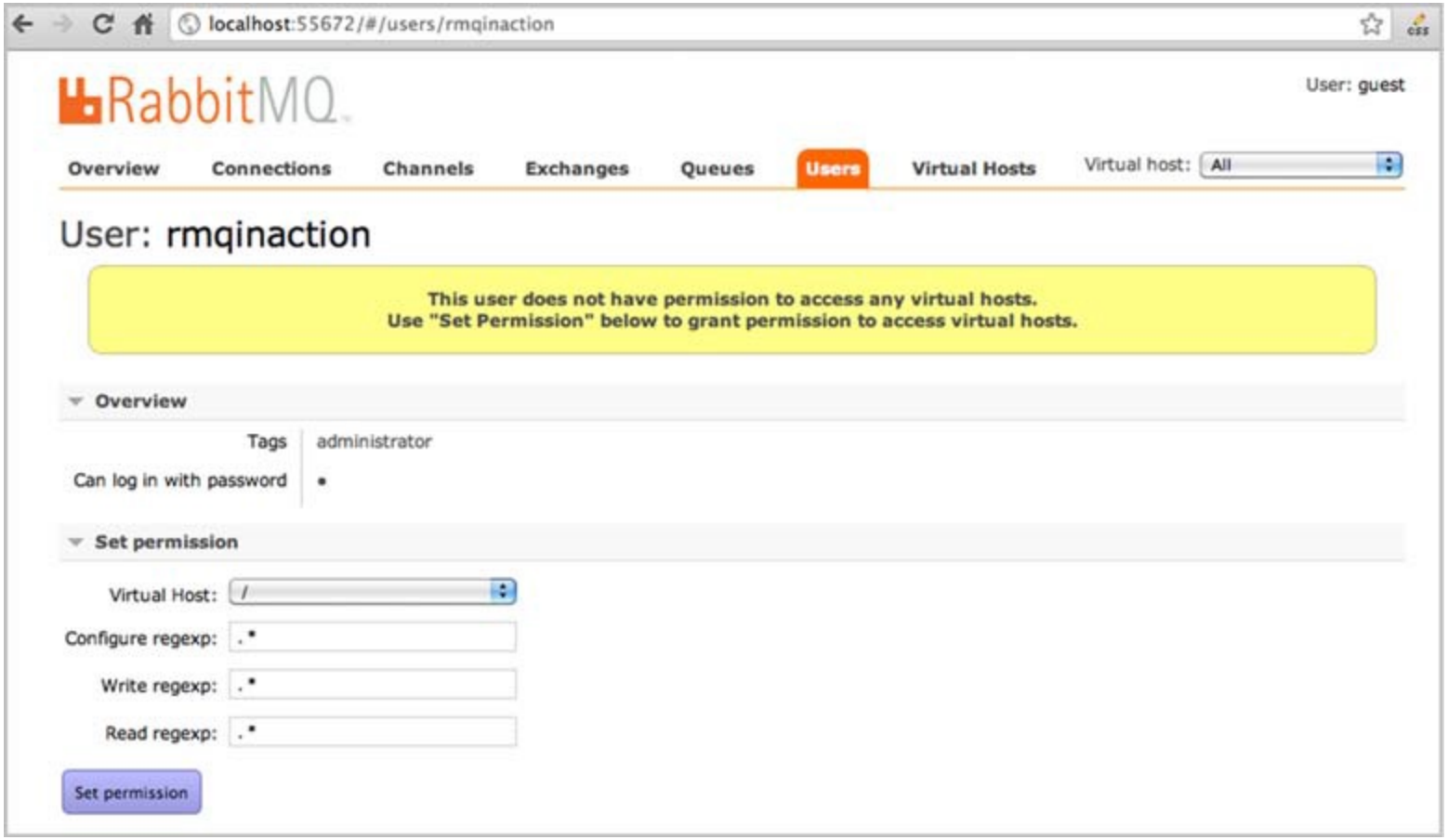
Managing users

- New user confirmation



Managing users

- Managing users' permissions



Lab: SSL connection