Absolutely — here's a **complete, ready-to-use F5 configuration guide** for your **RabbitMQ HTTPS (SSL Bridging)** setup, including all profiles, health monitor, pool, and virtual server definitions.

This version is designed for **production use**, with clear roles, realistic defaults, and full compatibility with RabbitMQ's management HTTPS port (`15671`).

F5 BIG-IP — RabbitMQ HTTPS SSL Bridging Configuration

Goal

- Expose a **public HTTPS VIP (port 443)** to clients.
- F5 decrypts client SSL (Client SSL Profile).
- F5 re-encrypts traffic to backend RabbitMQ nodes over port **15671** (Server SSL Profile).
- F5 performs **health checks** via `/api/overview` expecting HTTP `200 OK`.

---

1 Create the SSL Profiles

Client SSL Profile

Handles HTTPS from **client → F5**

```
tmsh create ltm profile client-ssl clientssl_rabbitmq \
    defaults-from clientssl \
    cert-key-chain add { rabbitmq_cert { certificate example.crt key
example.key chain ca-bundle.crt } } \
    ciphers "TLSv1.3:TLSv1.2:!RC4:!3DES:!aNULL" \
    options { no-sslv3 no-tlsv1 no-tlsv1_1 }
```

GUI Equivalent:

- Go to **Local Traffic → Profiles → SSL → Client → Create**
- Name: `clientssl_rabbitmq`
- Parent Profile: `clientssl`
- Attach certificate/key/chain
- Disable SSLv3/TLSv1.0
- Save

Server SSL Profile

Handles HTTPS from **F5 → RabbitMQ nodes**

```
tmsh create ltm profile server-ssl serverssl_rabbitmq \
    defaults-from serverssl \
    server-ssl-profile true \
    authenticate name trusted \
    ciphers "TLSv1.3:TLSv1.2:!RC4:!3DES:!aNULL"
```

(Optional: If you want to verify backend certificates, upload your CA bundle and use `ca-file` option.)

GUI Equivalent:

- **Local Traffic → Profiles → SSL → Server → Create**
- Name: `serverssl_rabbitmq`
- Parent Profile: `serverssl`
- Enable "Server Authentication" only if RabbitMQ certs are valid/trusted.

❓ 2 Create the HTTPS Health Monitor

Checks `/api/overview` on port `15671` and expects `200`.

```
tmsh create ltm monitor https https_rabbitmq_monitor \
    send "GET /api/overview HTTP/1.1\r\nHost: localhost\r\nConnection:
close\r\n\r\n" \
    recv "200" \
    interval 5 timeout 16 \
    alias-service-port 15671
```

GUI Equivalent:

- **Local Traffic → Monitors → Create**
  - Name: `https_rabbitmq_monitor`
  - Type: `HTTPS`
  - Send String:

    ```
    GET /api/overview HTTP/1.1\r\nHost: localhost\r\
    nConnection: close\r\n\r\n
    ```

  - Receive String: `200`
  - Alias Service Port: `15671`
  - Secure: ☑ checked (use SSL)
  - Interval: `5`, Timeout: `16`

This ensures F5 only marks nodes **UP** if the RabbitMQ management API replies `200 OK`.

---

3 Create the Pool

Add your RabbitMQ nodes listening on port 15671.

```
tmsh create ltm pool rabbitmq_pool \
    monitor https_rabbitmq_monitor \
    members add { 10.0.0.11:15671 10.0.0.12:15671 10.0.0.13:15671 }
```

GUI Equivalent:

- **Local Traffic → Pools → Create**
  - Name: `rabbitmq_pool`
  - Health Monitor: `https_rabbitmq_monitor`
  - Load Balancing: `round-robin` (or `least-connections`)
  - Members:
    - `10.0.0.11:15671`
    - `10.0.0.12:15671`
    - `10.0.0.13:15671`

---

4 Create the Virtual Server (VIP)

Expose HTTPS on **port 443** to clients.

```
tmsh create ltm virtual vs_rabbitmq_https \
    destination 10.0.0.100:443 \
    ip-protocol tcp \
    profiles add { http clientssl_rabbitmq serverssl_rabbitmq } \
    pool rabbitmq_pool \
    source 0.0.0.0/0 \
    persist none \
    translate-address enabled \
    translate-port enabled
```

GUI Equivalent:

- **Local Traffic → Virtual Servers → Create**
  - Name: `vs_rabbitmq_https`
  - Destination Address: `10.0.0.100`
  - Service Port: `443`
  - Protocol: `TCP`
  - HTTP Profile: `http`
  - SSL Profile (Client): `clientssl_rabbitmq`
  - SSL Profile (Server): `serverssl_rabbitmq`

- Default Pool: `rabbitmq_pool`
- Source Address Translation: `Auto Map` (recommended)

This is your **HTTPS entry point** (port 443). F5 decrypts → processes → re-encrypts to backend 15671.

---

5 Verify the Configuration

Health Check

```
tmsh show ltm pool rabbitmq_pool members
```

→ Should show all members **up (green)** if `/api/overview` responds 200.

Client Test

```
curl -k https://10.0.0.100/api/overview
```

→ You should see RabbitMQ's JSON response (cluster info).

SSL Handshake

```
openssl s_client -connect 10.0.0.100:443
```

→ Verify the certificate presented by F5 (client-side SSL).

---

6 Architecture Recap

```
Client (HTTPS)
    |
    v
 [ F5 Virtual Server :443 ]
    |  (ClientSSL decrypt)
    |
    v
 HTTP routing / health checks / load balancing
    |
    v
 [ F5 Pool → RabbitMQ nodes :15671 ]
    |  (ServerSSL re-encrypt)
    v
 RabbitMQ HTTPS endpoint (/api/overview)
```

Final Configuration Summary

Component	Name	Port	Role
Client SSL Profile	`clientssl_rabbitmq`	443	Decrypt client HTTPS
Server SSL Profile	`serverssl_rabbitmq`	156	Re-encrypt to backend

Component	Name	Port	Role
		71	
Health Monitor	`https_rabbitmq_monitor`	156	Check `/api/overview`
		71	
Pool	`rabbitmq_pool`	156	RabbitMQ backend nodes
		71	
Virtual Server	`vs_rabbitmq_https`	443	Public HTTPS endpoint

Would you like me to provide an **Ansible playbook** version of this configuration (using `bigip_*` modules) — so you can automate deployment directly to F5?