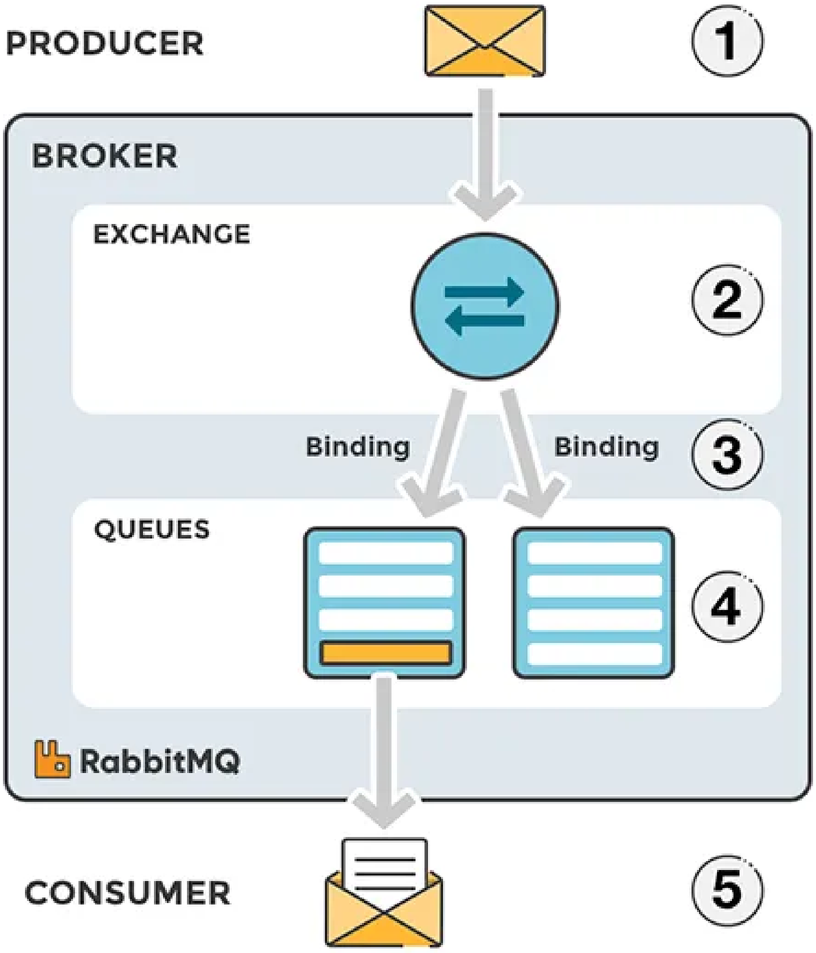
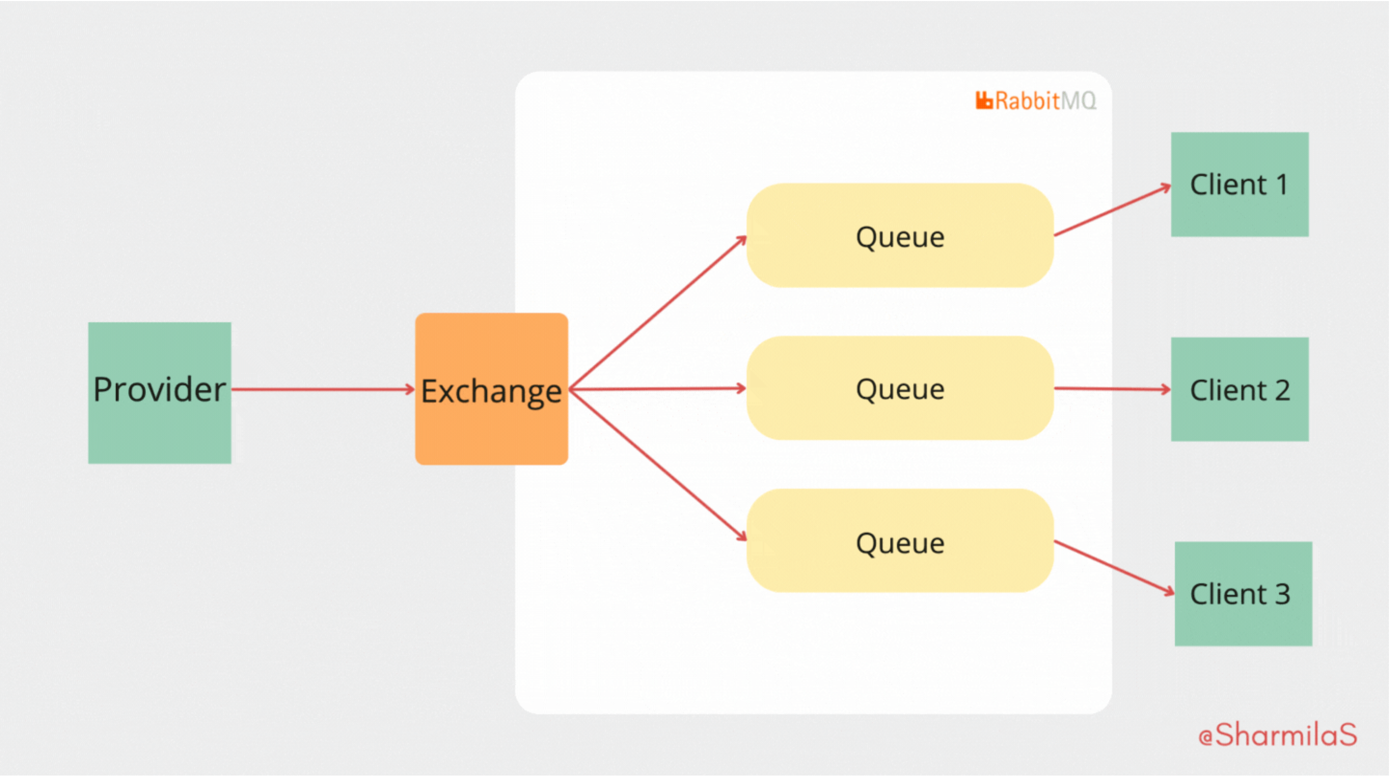
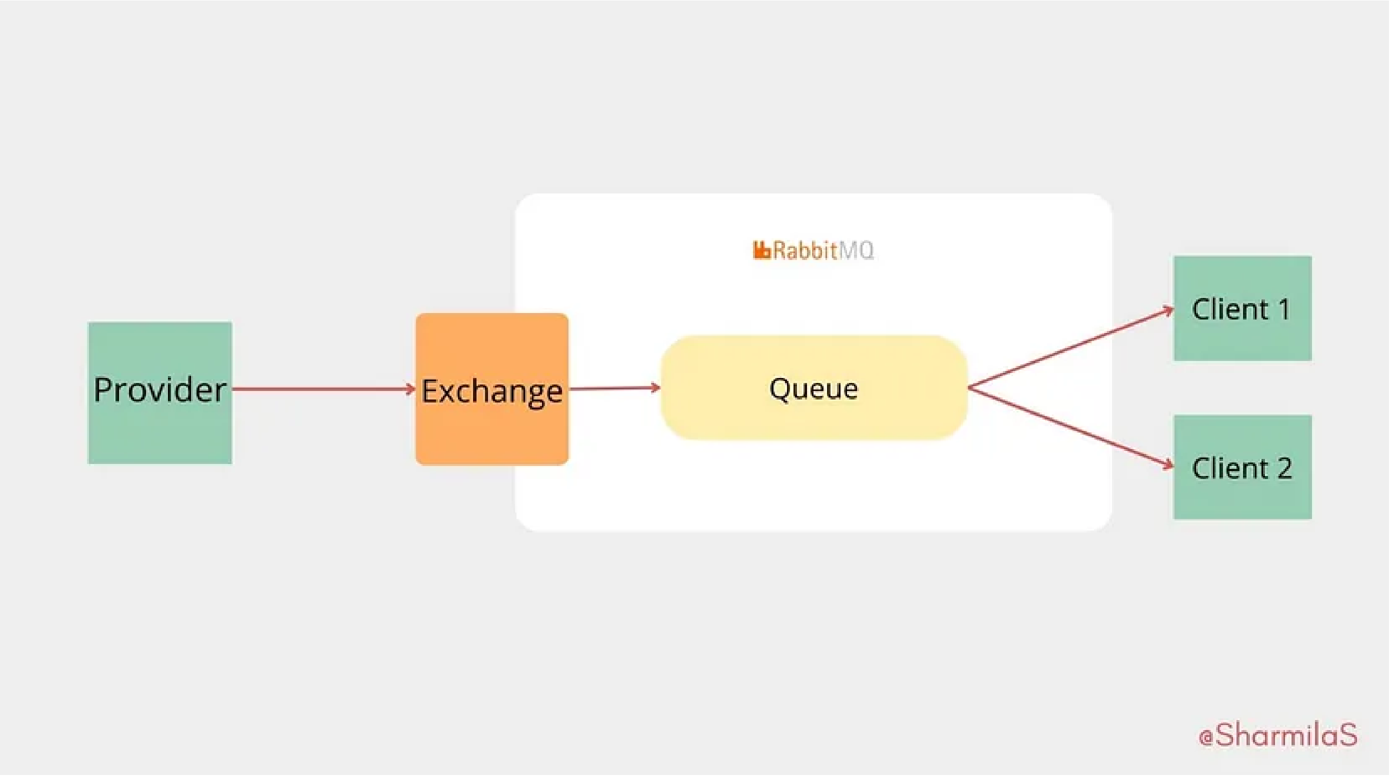
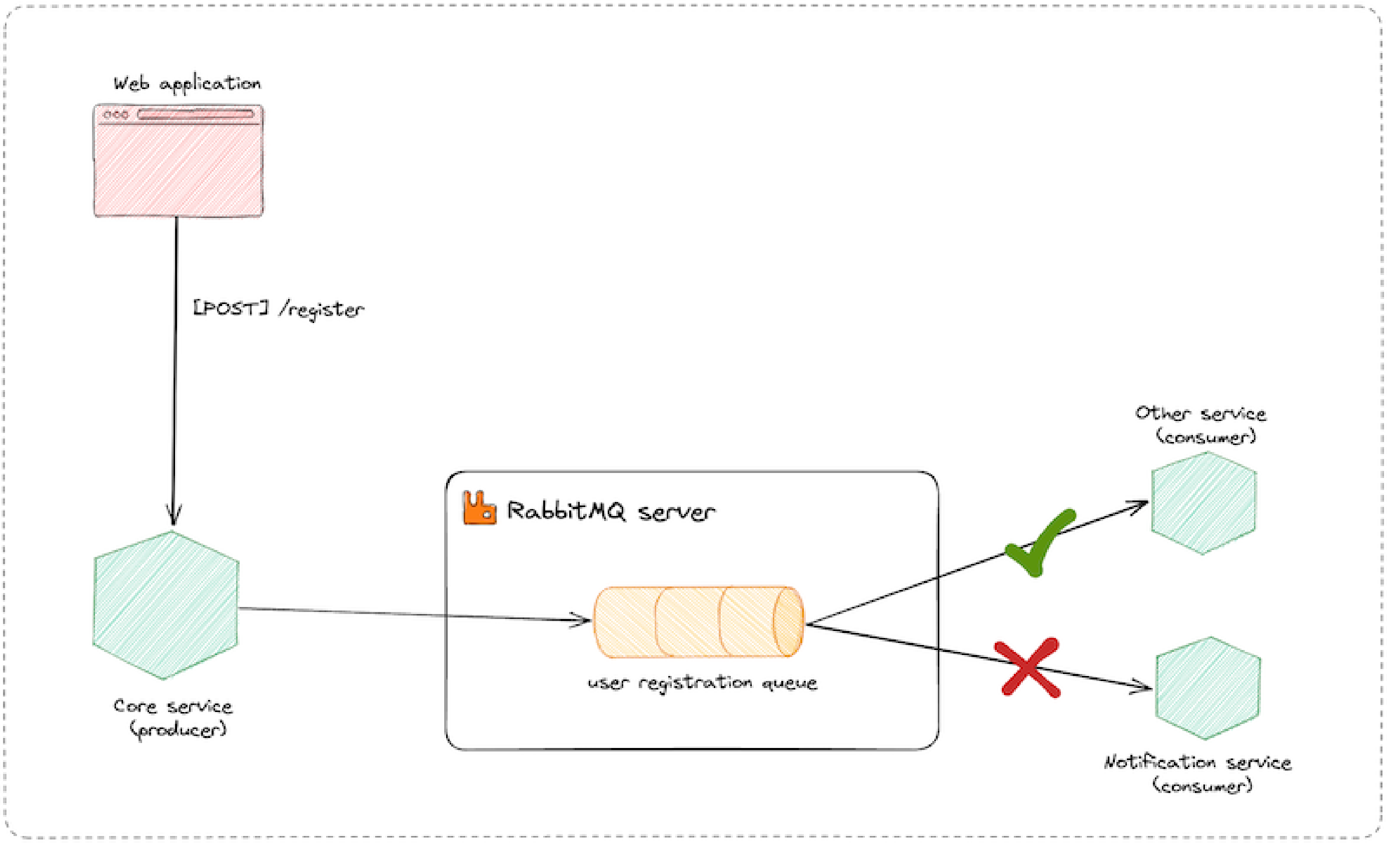
Node Rabbit MQ

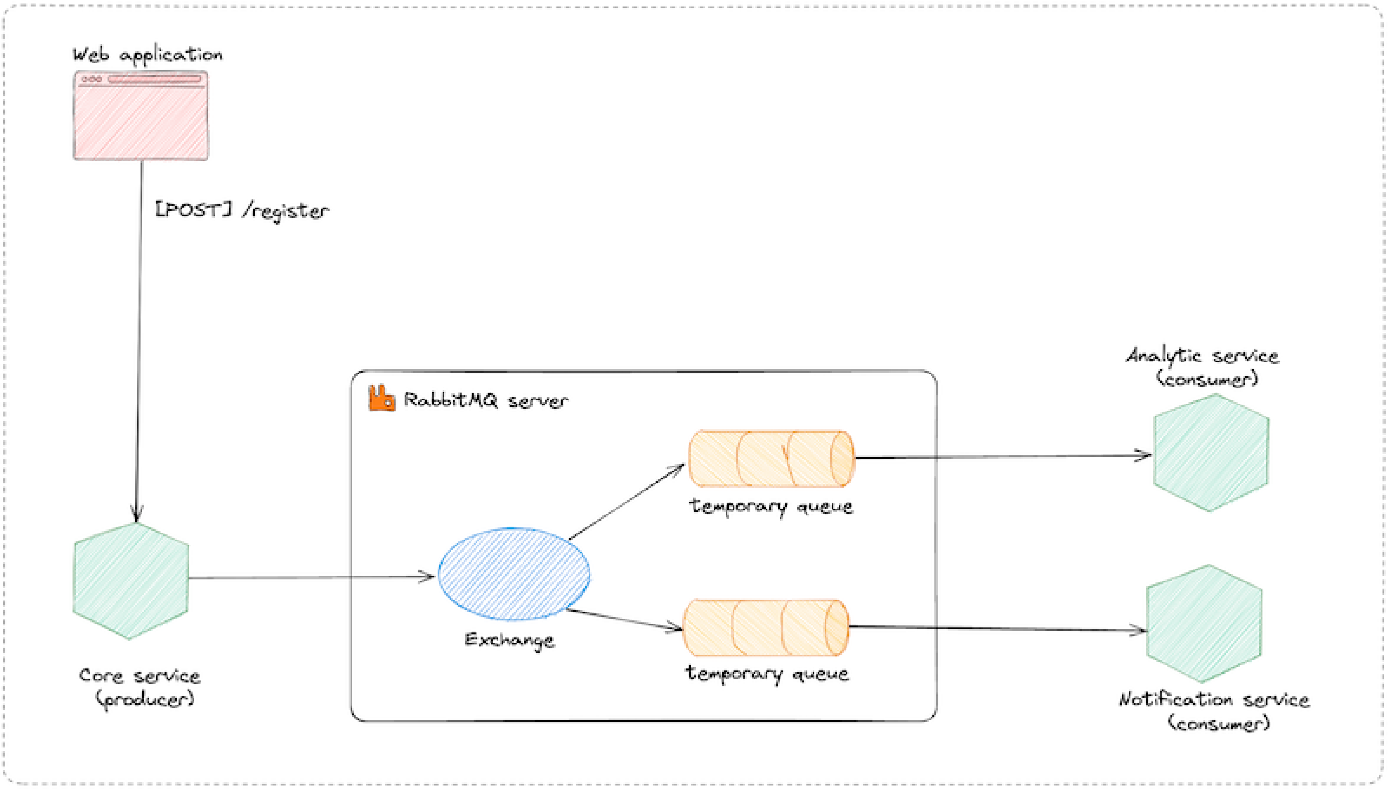
https://github.com/tericcabrel/blog-tutorials/tree/main/node-rabbitmq?ref=blog.tericcabrel.com




RabbitMQ
Provider
Exchange
Queue
Queue
Queue
Client 1
Client 2
Client 3
@SharmilaS


RabbitMQ
Provider
Exchange
Queue
Client 1
Client 2
@SharmilaS





```
export CLOUDAMQP_URL=amqp://**user**:**password**:**url**
```

```
var amqp = require('amqplib/callback_api');
var amqpConn = null;
```

```
function whenConnected() {
startPublisher();
startWorker();
}
```

```
function start() {
amqp.connect(process.env.CLOUDAMQP_URL + "?heartbeat=60", function(err, conn) {
if (err) {
console.error("[AMQP]", err.message);
return setTimeout(start, 1000);
}
conn.on("error", function(err) {
if (err.message !== "Connection closing") {
console.error("[AMQP] conn error", err.message);
}
});
conn.on("close", function() {
console.error("[AMQP] reconnecting");
return setTimeout(start, 1000);
});
console.log("[AMQP] connected");
amqpConn = conn;
whenConnected();
});
}
```

```
var pubChannel = null;
var offlinePubQueue = [];
function startPublisher() {
amqpConn.createConfirmChannel(function(err, ch) {
if (closeOnErr(err)) return;
ch.on("error", function(err) {
console.error("[AMQP] channel error", err.message);
});
ch.on("close", function() {
console.log("[AMQP] channel closed");
});
pubChannel = ch;
while (true) {
var [exchange, routingKey, content] = offlinePubQueue.shift();
publish(exchange, routingKey, content);
}
});
}
```

```
function publish(exchange, routingKey, content) {
try {
pubChannel.publish(exchange, routingKey, content, { persistent: true },
function(err, ok) {
if (err) {
console.error("[AMQP] publish", err);
offlinePubQueue.push([exchange, routingKey, content]);
pubChannel.connection.close();
}
});
} catch (e) {
console.error("[AMQP] publish", e.message);
offlinePubQueue.push([exchange, routingKey, content]);
}
}
```

```
// A worker that acks messages only if processed successfully
function startWorker() {
amqpConn.createChannel(function(err, ch) {
if (closeOnErr(err)) return;
ch.on("error", function(err) {
console.error("[AMQP] channel error", err.message);
});
ch.on("close", function() {
console.log("[AMQP] channel closed");
});
ch.prefetch(10);
ch.assertQueue("jobs", { durable: true }, function(err, _ok) {
if (closeOnErr(err)) return;
ch.consume("jobs", processMsg, { noAck: false });
console.log("Worker is started");
});
});
}
```

```
function processMsg(msg) {
work(msg, function(ok) {
try {
if (ok)
ch.ack(msg);
else
ch.reject(msg, true);
} catch (e) {
closeOnErr(e);
}
});
}
```

```
function work(msg, cb) {
console.log("PDF processing of ", msg.content.toString());
cb(true);
}
```

```
function closeOnErr(err) {
if (!err) return false;
console.error("[AMQP] error", err);
amqpConn.close();
return true;
}
```

```
setInterval(function() {
publish("", "jobs", new Buffer("work work work"));
}, 1000);

start();
```

https://medium.com/@rafael.guzman/how-to-consume-publish-rabbitmq-message-in-nodejs-cb68b5a6484c

```
npm install amqplib --save
```

```
export CLOUDAMQP_URL=amqp://**user**:**password**:**url**
```

```
let amqpConn = null;
module.exports = {
InitConnection: (fnFinish) => {
// Start connection with Rabbitmq
amqp.connect(process.env.CLOUDAMQP_URL, (err, conn) => {
// If connection error
if (err) {
console.error("[AMQP]", err.message);
return setTimeout(this, 1000);
}
conn.on("error", function(err) {
console.log("ERROR", err);
if (err.message !== "Connection closing") {
console.error("[AMQP] conn error", err.message);
}
});
conn.on("close", function() {
// Reconnect when connection was closed
console.error("[AMQP] reconnecting");
return setTimeout(() => { module.exports.InitConnection(fnFinish) }, 1000);
});
// Connection OK
console.log("[AMQP] connected");
amqpConn = conn;
// Execute finish function
fnFinish();
});
}
};
```

```
const rabbitmqLib = require('rabbitmq-initconnection.js');
// InitConnection of rabbitmq
rabbitmqLib.InitConnection(() => {
console.log("Connection with Rabbitmq was successful);
});
```

```
const amqp = require("amqplib");
async function connect() {
try {
const connection = await amqp.connect("amqp://localhost:5672");
const channel = await connection.createChannel();
await channel.assertQueue("number");
channel.consume("number", message => {
const input = JSON.parse(message.content.toString());
console.log(`Received number: ${input.number}`);
channel.ack(message);
});
console.log(`Waiting for messages...`);
} catch (ex) {
console.error(ex);
}
}
connect();
```