#### **RabbitMQ Core Concepts Overview**

1. **Exchanges**:
   * Exchanges are where producers send messages. The exchange then routes these messages to one or more queues depending on the type of exchange and the binding rules.
   * **Types of Exchanges**:
     1. **Direct**: Routes messages to queues based on exact matching between the routing key and the binding key.
     2. **Topic**: Routes messages based on pattern matching between routing key and binding key (e.g., \*.log).
     3. **Fanout**: Broadcasts all messages to all bound queues, ignoring routing keys.
     4. **Headers**: Routes messages based on message headers instead of routing keys.
2. **Queues**:
   * A buffer that stores messages until consumed by a worker.
   * Messages stay in queues until acknowledged by consumers, which means they are persistent until the worker processes them or until they expire.
3. **Bindings**:
   * Bindings link exchanges to queues. A binding can include a routing key, which defines the rules for routing messages.
4. **Routing Keys**:
   * A string used by exchanges to determine where to send messages. Exchanges like **direct** and **topic** use routing keys to filter messages to specific queues.
5. **Message Flow**:
   * **Producer → Exchange → Queue → Consumer**:
     1. The producer sends a message to the exchange.
     2. The exchange looks at the routing key and forwards the message to the appropriate queue(s).
     3. The consumer retrieves the message from the queue and processes it.
6. **Message Acknowledgment**:
   * Ensures that messages are processed successfully. If the consumer does not acknowledge a message, RabbitMQ can re-deliver the message.
   * Acknowledgments are crucial for message reliability and preventing message loss.
7. **Message Persistence**:
   * Persistent messages are saved to disk and survive broker restarts. You can mark messages as persistent by setting the delivery mode to 2 when publishing.

### **2. Hands-on (2 hours)**

#### **Step 1: Setting Up RabbitMQ with Docker**

* **Install Docker** (if not already installed):
  + For Windows/Mac: Download from Docker's official website.
  + For Linux: Use the package manager of your distribution (e.g., apt-get install docker.io).

**Pull RabbitMQ Docker Image**:  
bash  
Copy code  
docker pull rabbitmq:management

* This image comes with the management plugin pre-installed.

**Run RabbitMQ Container**:  
bash  
Copy code  
docker run -d --name rabbitmq -p 5672:5672 -p 15672:15672 rabbitmq:management

* + -d: Run in detached mode.
  + --name rabbitmq: Name the container.
  + -p 5672:5672: Map the RabbitMQ port.
  + -p 15672:15672: Map the RabbitMQ Management UI port.
* **Access RabbitMQ Management Console**:
  + Open a browser and navigate to http://localhost:15672.
  + Login with the default credentials (guest / guest).

#### **Step 2: Configuring RabbitMQ Plugins**

* **Enable Shovel Plugin**:

Within the container, run:  
bash  
Copy code  
docker exec -it rabbitmq bash

rabbitmq-plugins enable rabbitmq\_shovel rabbitmq\_shovel\_management

exit

* **Enable Federation Plugin**:

Similarly, within the container, run:  
bash  
Copy code  
docker exec -it rabbitmq bash

rabbitmq-plugins enable rabbitmq\_federation rabbitmq\_federation\_management

exit

* **Verify Plugins**:
  + Go back to the Management UI (http://localhost:15672).
  + Navigate to the "Admin" tab → "Plugins" section to confirm that Shovel, Federation, and other plugins are enabled.

#### **Step 3: Creating and Managing RabbitMQ Components**

* **Create an Exchange**:
  + In the Management UI, go to the "Exchanges" tab and create a new exchange:
    - Name: my\_exchange
    - Type: direct
* **Create a Queue**:
  + In the "Queues" tab, create a new queue:
    - Name: my\_queue
* **Bind the Queue to the Exchange**:
  + In the "Exchanges" tab, select my\_exchange.
  + Scroll down to the "Bindings" section and bind my\_queue with a routing key, e.g., my\_routing\_key.
* **Send and Receive a Test Message**:
  + **Send**: In the "Exchanges" tab, publish a message to my\_exchange with my\_routing\_key.
  + **Receive**: Go to the "Queues" tab, click on my\_queue, and get messages.
* **Message Acknowledgment and Persistence**:
  + To ensure messages are acknowledged, use the basic.ack command in the consumer code.
  + To make messages persistent, set the delivery\_mode to 2 when publishing messages.