1. Email Schedule

Objective: Send four emails at specific intervals for each software release.

Steps: We need an scheduling tool for this task. Amazon EventBridge Scheduler is suitable as this tool allows for recurring schedules.

2. Content Generation

Objective: Generate email content that includes version number, release notes, fixed bugs, and new features.

Steps:

1. Fetch relevant data using the Jira API.
2. Generate Email Content:
    - OpenAI API is a good tool for automatic content generation .
    - Example prompt: "Generate an email informing clients about the upcoming software release. Include the version number, release notes, fixed bugs, and new features.
3. Extract and format the response from OpenAI for each email.

3. Data Retrieval

Objective: Retrieve relevant release information from Jira.

Steps:

1. Set up JIRA API access e.g keys, authentication tokens.
2. Ensure the automated system has access to release data.
3. Validate the obtained data from Jira before sending them to OpenAI for content generation.

4. Email Sending

Objective: Automatically sends email at the required intervals.

Steps:

1. Select an email service provider such as SendGrid or Amazon Simple Email Service (SES). If we are going with EventBridge as the scheduler, SES should be the better option because they are both on AWS, with detailed documentation.
2. Set up email authentication e.g DKIM and SPF.
3. Create email templates, which are available via the CreateTemplate API.
4. Use the provider's API to send emails. In case where we need to send emails to multiple recipients, it is best to set up the system so that each email will be sent to each recipient one at a time. The clients probably do not need to know each other's details, and the emails would be more personalised, potentially improve client relations.

SES also supports sending emails to mutiple recipients via the SendEmail API.

5. Error Handling

Objective: Handle potential errors during the automation process.

Steps:

1. API request failures:
    - Implement retry with backoff pattern to avoid overloading APIs and being restricted from using them.
    - Logs errors for review and debugging.
2. Email sending failures:
    - Monitor email bounce rate/failures to deliver.
    - Ensure email content follows email regulations of the recipient's locality e.g the Spam Act 2003 for Australia-based clients.

6. Testing and Monitoring

Objective: Ensures the automated system works correctly – emails are sent as scheduled.

Steps:

1. Unit testing:
    - Write tests for every single module of the system e.g retrieval from JIRA API, content generation via OpenAI API, scheduling jobs via EventBridge, etc...
    - Set up mock APIs to simulate API responses.
2. Integration testing: Test the entire workflow in a staging environment.
3. Monitor: Since we are using SES, we can monitor our sending activity using events, metrics, and statistics obtainable from the SES API. We can use CloudWatch to view these metrics as ordered time-series data sets. CloudWatch can also be configured to provide more fine-grained metrics as needed.

7. Scalability

Objective: Scale the automation to handle multiple software releases and clients simultaneously.

Steps:

1. Containerisation: I would use Docker to containerise the applications, to ensure the system work correctly across different environments i.e simplifying deployments. Kubernette

2. Microservice architecture: Scheduling, data retrieval, content generation , email sending should be handled separately by different modules of the system. This architecture allows us to scale individual components more easily, as needed.