

### 3.0 Objectives

Too many emails? Need content in a different format? Want notifications from tools that don't generate them?

- Workflow vs. Automation - is there a difference?
- Post-its – the old fashioned workflow controller
- IFTTT & Zapier - make life easier
- Node-Red - it's free, it's powerful, it's sometimes easy
- n8n, prismatic, roll-your own & good old Claude code
- Google opal - a game changer?

### Learning Outcomes

- Basic understanding of business workflow involving humans + machines
- Practical experience of common free tools for experimentation and prototyping
- Understanding No-Code, Scripted and Coded tools – where to use what, when

Published

Assign To

Edit

Student Pairs - assignment a3.1

|        |
|--------|
| 1, 12  |
| 13, 11 |
| 14, 10 |
| 15, 9  |
| 16, 8  |
| 2, 7   |
| 3, 6   |
| 4, 5   |

You are 007's handler. 007 is on a mission in Geneva and is asleep on the 3rd floor of the Hotel d'Angleterre. Write a workflow that details the triggers, processes, events & logs that 007 must take to:

- 1

Wake up & wash
- 2

Dress & descend to the terraced café
- 3

Receive a magazine from the local agent
- 4

Board the ferry to cross the lake to France.

→ think about 007's day job and anything special that might involve

→ it's 2025, so WiFi, mobile phones and modern tools exist

→ write your workflow on a SINGLE post-it page. 007 is not good with long documents

→ when you're done, post your note with your badge numbers on the wall

Points

0

Submitting

Nothing

| Due | For      | Available from | Until |
|-----|----------|----------------|-------|
| -   | Everyone | -              | -     |

# Workflow Vocabulary

## Essential terms for live media production workflows

---

### Node

#### Noun

A discrete functional unit within a workflow that performs a specific operation on data or media. Nodes receive inputs, process information, and produce outputs that can be connected to other nodes.

*Example: A transcoding node converts incoming video from one format to another before passing it downstream.*

### Flow

#### Noun

The complete path of data or media through a series of connected nodes, representing the entire workflow from input to output. A flow defines how information moves and transforms through the system.

*Example: The ingest flow captures raw camera feeds, applies color correction, and routes them to the production switcher.*

### Trigger

#### Verb / Noun

An action or condition that initiates a workflow or activates a specific node. Triggers can be time-based, event-driven, or manual, causing downstream processes to execute.

*Example: The director's command triggers the instant replay workflow to capture and queue the last 30 seconds of footage.*

# Event

## Noun

A significant occurrence or change in state within the workflow system that can be detected, logged, and used to trigger subsequent actions. Events carry metadata about what happened and when.

*Example: A "signal loss" event from a camera feed automatically switches to a backup source.*

# Process

## Verb / Noun

The act of transforming or manipulating media content or data as it moves through the workflow. As a noun, it refers to a specific transformation operation or the overall workflow execution.

*Example: The audio process normalizes levels and applies compression before the signal reaches the broadcast encoder.*

# Pipeline

## Noun

A linear sequence of processing stages where the output of one stage becomes the input of the next. Pipelines enable systematic, predictable media transformations with clear data flow.

*Example: The graphics pipeline renders lower-thirds, overlays sponsor logos, and composites them with the main video feed.*

# Artifact

## Noun

A discrete piece of media content or data file created as a result of workflow processing. Artifacts are tangible outputs that can be stored, referenced, or used as inputs for other workflows.

*Example: Each recorded segment generates artifacts including the master video file, proxy files, and associated metadata.*

# Queue

## Noun / Verb

A holding area where media items or tasks wait in sequence to be processed. Queues manage workload by buffering inputs and ensuring orderly processing when resources become available.

*Example: Instant replay clips are queued for review, allowing the operator to select and play them in the desired order.*

## Log

Noun / Verb

A chronological record of events, actions, and system states within the workflow. Logging captures operational data for monitoring, debugging, compliance, and performance analysis.

*Example: The workflow logs all incoming timecode breaks, allowing engineers to diagnose synchronization issues post-broadcast.*

## Route

Verb / Noun

The act of directing media signals or data from a source to one or more destinations based on logic, conditions, or manual selection. Routing determines the path content takes through the system.

*Example: Based on content type, the system routes sports feeds to the stadium displays and news feeds to the control room monitors.*

## Inject

Verb

To insert data, media, or signals into an existing workflow or stream at a specific point. Injection allows dynamic addition of content without disrupting the continuous flow.

*Example: The operator injects emergency broadcast alerts into all active output streams simultaneously.*

## Deploy

Verb

To activate and make a workflow or node configuration operational in the production environment. Deployment transitions workflows from design or testing states to live, active execution.

*Example: After testing the new graphics workflow in staging, the team deploys it to the live broadcast system for tonight's game.*

Built with care for the media production community | [Mr MXF](#) 

Published

Assign To

Edit

Student Pairs - assignment a3.2

|        |
|--------|
| 1, 11  |
| 12, 10 |
| 13, 9  |
| 14, 8  |
| 15, 7  |
| 16, 6  |
| 2, 5   |
| 3, 4   |

Here is a sample workflow using IFTTT: <https://ifttt.com/applets/XwJXdnj4-save-all-of-your-liked-videos-in-a-spreadsheet> ➡

- 1
- Log into / create your IFTTT Account
- 2
- Follow the instructions and grant permissions for YouTube and Google
- 3
- Try it!
- 4
- Now recreate your own similar workflow from scratch, here are some ideas:
  - thank you email someone who subscribes to your channel
  - get an email if a new video appears on your favourite topic e.g. "Springboard Diving", "Stormtrooper Vlog", "Fat cat fails"
  - get your phone to notify you via the IFTTT app if you need to wear a coat tomorrow
  - get your phone to notify you via the IFTTT app when Spotify releases a new podcast you like
  - get your phone to play "I feel good" every time you start charging it

Points0

SubmittingNothing

| Due | For      | Available from | Until |
|-----|----------|----------------|-------|
| -   | Everyone | -              | -     |

# Composable Processes in Workflows

Building reliable, maintainable systems through quantifiable design principles

**Composable processes** are self-contained units of work that can be combined, reused, and orchestrated to create complex workflows. By emphasizing quantifiable properties, we can build systems that are not only functional but measurable, predictable, and sustainable over time.

This guide focuses on five critical properties: **dependencies**, **observability**, **stability**, **maintainability**, and **modularity**.

## Core Principles

### 1. Dependencies: Minimize and Make Explicit

Every dependency is a potential point of failure and a constraint on reusability. The goal is to make dependencies explicit, minimal, and measurable.

#### Quantifiable Metrics:

**Dependency Count** Number of external services, libraries, or processes required

**Coupling Score** Ratio of external calls to internal operations

**Dependency Depth** Maximum chain length in the dependency graph

**Version Lock Ratio** Percentage of dependencies with pinned versions

#### Design Techniques:

- **Dependency Injection:** Pass dependencies explicitly rather than hardcoding them
- **Interface Segregation:** Depend on minimal interfaces, not concrete implementations
- **Version Contracts:** Define clear API contracts and use semantic versioning

- **Dependency Inversion:** High-level processes should not depend on low-level details

## 2. Observability: Measure Everything That Matters

If you can't measure it, you can't improve it. Observability enables debugging, optimization, and confident decision-making.

### Quantifiable Metrics:

**Execution Time** P50, P95, P99 latencies for process completion

**Success Rate** Percentage of successful completions vs. failures

**Resource Utilization** CPU, memory, disk, network usage patterns

**Error Rate** Frequency and types of errors encountered

**Instrumentation Coverage** Percentage of code paths with logging/metrics

### Design Techniques:

- **Structured Logging:** Use consistent, parseable log formats with context
- **Distributed Tracing:** Track request flows across process boundaries
- **Health Checks:** Expose readiness and liveness endpoints
- **Metrics Export:** Emit metrics in standard formats (Prometheus, StatsD)
- **Correlation IDs:** Propagate unique identifiers through the system

## 3. Stability: Design for Failure

Processes will fail. Design systems that degrade gracefully and recover automatically.

### Quantifiable Metrics:

**MTBF** Mean Time Between Failures

**MTTR** Mean Time To Recovery

**Error Budget** Acceptable failure rate over time period

**Circuit Breaker Trips** Frequency of protective mechanisms activating

Retry Success Rate

Percentage of operations that succeed after retry

#### Design Techniques:

- **Idempotency:** Ensure operations can be safely retried without side effects
- **Circuit Breakers:** Prevent cascading failures by failing fast
- **Timeouts:** Set explicit deadlines for all external calls
- **Graceful Degradation:** Provide reduced functionality rather than complete failure
- **Bulkheads:** Isolate resources to prevent total system failure
- **Backpressure:** Signal upstream when capacity is exceeded

## 4. Maintainability: Optimize for Change

Code is read far more often than it's written. Prioritize clarity and ease of modification.

#### Quantifiable Metrics:

Cyclomatic Complexity

Number of independent paths through code

Code Churn

Frequency of changes to specific modules

Documentation Coverage

Ratio of documented vs. undocumented components

Test Coverage

Percentage of code exercised by tests

Time to Onboard

Days for new developer to make productive changes

#### Design Techniques:

- **Self-Documenting Code:** Use clear names that explain intent
- **Single Responsibility:** Each process does one thing well
- **Comprehensive Testing:** Unit, integration, and end-to-end tests
- **README-Driven Development:** Document the "why" before the "how"
- **Code Comments:** Explain decisions, not mechanics
- **Consistent Patterns:** Use established patterns across the codebase

## 5. Modularity: Build with Boundaries

Well-defined boundaries enable parallel development, testing, and deployment.

#### Quantifiable Metrics:

Interface Stability

Frequency of API contract changes

Module Size

Lines of code per module (target: 200-500)

Reuse Factor

Number of consumers for each module

Deployment Independence

Percentage of modules deployable separately

API Surface Area

Number of public methods/endpoints

### Design Techniques:

- **API-First Design:** Define interfaces before implementation
- **Information Hiding:** Expose only what consumers need
- **Versioned Contracts:** Allow evolution without breaking changes
- **Separation of Concerns:** Isolate data, logic, and presentation
- **Minimal Public APIs:** Reduce the surface area for breaking changes

## Pros and Cons

### Benefits of Composable Processes

- **Reusability:** Write once, use in multiple workflows
- **Testability:** Isolated processes are easier to test
- **Parallel Development:** Teams can work independently
- **Incremental Deployment:** Update components without full redeployment
- **Clear Ownership:** Each process has a defined purpose and owner
- **Easier Debugging:** Isolate problems to specific components
- **Scalability:** Scale individual processes based on demand
- **Technology Diversity:** Use the best tool for each job

### Challenges and Trade-offs

- **Increased Complexity:** More moving parts to coordinate
- **Network Overhead:** Inter-process communication has latency
- **Distributed Debugging:** Harder to trace issues across boundaries
- **Data Consistency:** Maintaining consistency across processes is difficult
- **Deployment Complexity:** More components to deploy and monitor
- **Over-Engineering Risk:** Can lead to unnecessary abstraction
- **Learning Curve:** Team needs to understand the architecture

- **Operational Overhead:** More infrastructure to manage

## Managing Complexity: Decision Framework

Not every problem requires composable processes. Use this framework to make informed decisions:

### When to Decompose

- **Different Scale Characteristics:** Components have different scaling needs
- **Independent Evolution:** Parts change at different rates
- **Clear Boundaries:** Natural seams exist in the domain
- **Team Boundaries:** Different teams own different capabilities
- **Technology Mismatch:** Different components benefit from different tech stacks
- **Reuse Opportunities:** Component can be used in multiple contexts

### When to Keep Together

- **Tight Coupling:** Components share significant state or logic
- **High Communication:** Constant back-and-forth between components
- **Transactional Boundaries:** Operations must be atomic
- **Early Stage:** Requirements are still evolving rapidly
- **Simple Workflows:** Overhead exceeds benefits
- **Performance Critical:** Latency budget doesn't allow network calls

### Start Simple, Evolve Gradually

1. **Monolith First:** Begin with a simple, unified system
2. **Identify Pain Points:** Let real problems guide decomposition
3. **Extract Strategically:** Pull out components based on need, not speculation
4. **Measure Impact:** Use quantifiable metrics to validate improvements
5. **Iterate:** Continuously refine boundaries based on operational data

## Practical Guidelines

### 1. Define Success Criteria First

Before building, establish measurable goals:

- What is the acceptable latency? (e.g., P95 < 100ms)

- What is the target availability? (e.g., 99.9% uptime)
- What is the acceptable failure rate? (e.g., < 0.1% error rate)
- What resources are available? (e.g., 2 CPU cores, 4GB RAM)

## 2. Instrument from Day One

Build observability into every process:

- Log structured data at appropriate levels
- Emit metrics for duration, success rate, and resource usage
- Include correlation IDs in all logs and traces
- Expose health endpoints for monitoring systems

## 3. Design for Failure

Assume everything will fail eventually:

- Implement retries with exponential backoff
- Set timeouts on all external calls
- Use circuit breakers to prevent cascade failures
- Ensure operations are idempotent where possible
- Validate inputs at boundaries

## 4. Document Decisions

Capture the "why" behind choices:

- Why this architecture over alternatives?
- What trade-offs were made?
- What assumptions underpin the design?
- What are the operational runbooks?

## 5. Prioritize Reliability Over Performance

Focus on correctness and stability first:

- Prefer simple, understandable code over clever optimizations
- Optimize only after measuring actual bottlenecks
- Choose battle-tested libraries over cutting-edge alternatives
- Build automated tests before optimizing

# Conclusion

---

Composable processes enable building complex, reliable workflows from simple, well-understood components. By focusing on **quantifiable properties** and making deliberate trade-offs, we can create systems that are:

- **Observable:** We can see what's happening
- **Stable:** They handle failures gracefully
- **Maintainable:** They're easy to understand and modify
- **Modular:** They can evolve independently
- **Loosely Coupled:** They minimize unnecessary dependencies

Remember: *Start simple, measure constantly, and evolve based on real needs rather than anticipated ones.*

## Email to SMS Workflow: Tool Comparison

A comprehensive guide to implementing automated email-to-SMS notifications using four different workflow automation platforms

### The Challenge

You need to receive SMS notifications when important emails arrive. This is a common workflow automation task that can have various strengths and trade-offs.

**Our Workflow:** Email arrives → Extract key information → Send SMS notification to your phone

### Quick Comparison

#### Zapier

**Best for:** Non-technical users who want quick setup

- **Cost:** Free tier available (100 tasks/month)
- **Setup Time:** 5-10 minutes
- **Coding Required:** None
- **Hosting:** Cloud-based (no infrastructure needed)

[View Zapier Guide →](#)

#### n8n

**Best for:** Users wanting free unlimited automation with some technical comfort

- **Cost:** Free self-hosted (unlimited workflows)
- **Setup Time:** 15-30 minutes
- **Coding Required:** Optional (JavaScript for advanced features)
- **Hosting:** Self-hosted or cloud trial

[View n8n Guide →](#)

## Node-RED

**Best for:** Developers who want full control and customization

- **Cost:** Free and open source
- **Setup Time:** 20-45 minutes
- **Coding Required:** Some JavaScript for logic
- **Hosting:** Docker container (self-hosted)

[View Node-RED Guide →](#)

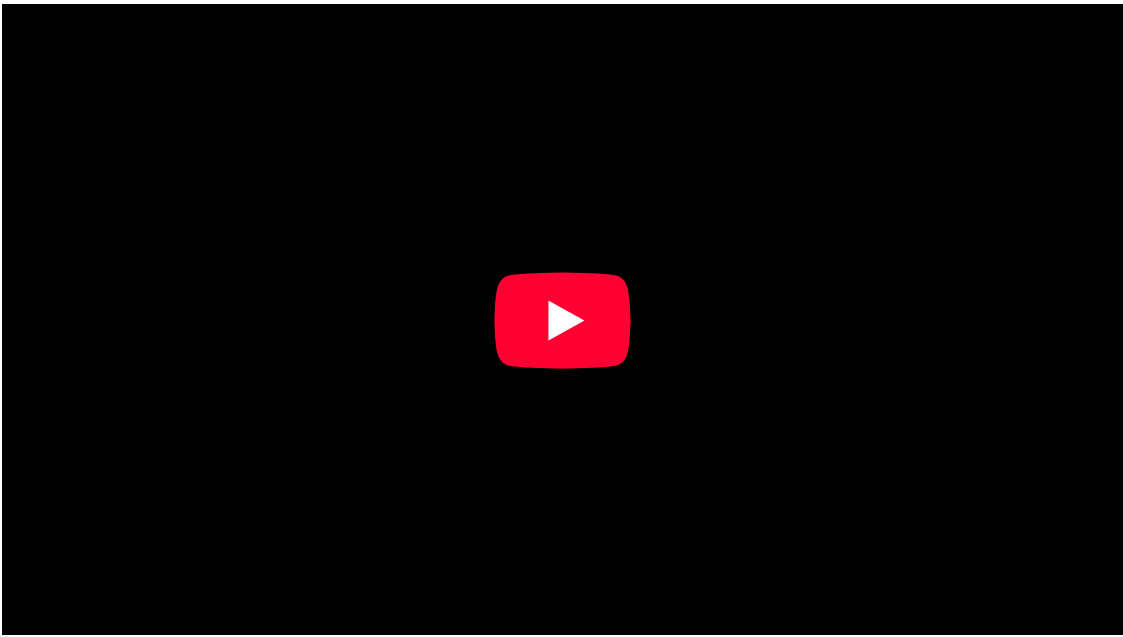
## Google Vertex AI

**Best for:** Enterprise users wanting AI-powered automation on Google Cloud

- **Cost:** Free tier + pay-as-you-go
- **Setup Time:** 20-35 minutes
- **Coding Required:** Minimal (webhook function only)
- **Hosting:** Google Cloud (serverless)

[View Vertex AI Guide →](#)

## opal.google



## Detailed Comparison

### Key Decision Factors

| Factor      | Zapier | n8n   | Node-RED |
|-------------|--------|-------|----------|
| Ease of Use | ★★★★★  | ★★★★★ | ★★★★     |

| Factor          | Zapier            | n8n    | Node-RED |
|-----------------|-------------------|--------|----------|
| Cost Efficiency | ★★ (limited free) | ★★★★★★ | ★★★★★★   |
| Flexibility     | ★★★               | ★★★★   | ★★★★★★   |
| Maintenance     | ★★★★★★ (zero)     | ★★★    | ★★       |
| Reliability     | ★★★★★★            | ★★★★   | ★★★      |

## Which Tool Should You Choose?

### Choose Zapier if you:

- Want the absolute fastest setup
- Don't want to write any code
- Need less than 100 automations per month
- Value simplicity over customization
- Don't want to manage any infrastructure

### Choose n8n if you:

- Want unlimited free automation
- Need a visual workflow builder
- Are comfortable with basic self-hosting
- Want more control than Zapier
- May need custom code occasionally

### Choose Node-RED if you:

- Are a developer or technically proficient
- Want complete control and customization
- Have Docker infrastructure available
- Need complex logic and transformations
- Want to integrate with IoT devices

### Choose Vertex AI if you:

- Want AI-powered email classification
- Need enterprise-grade Google Cloud infrastructure
- Already use Google Cloud Platform
- Want visual no-code agent builder
- Need advanced workflow features
- Value built-in testing and versioning

**Note on SMS Services:** All workflows require an SMS service provider. Popular options include:

- **Twilio:** Most popular, pay-as-you-go pricing, excellent documentation
- **MessageBird:** Good European coverage, competitive pricing
- **AWS SNS:** Very cheap for low volumes, requires AWS account
- **Plivo:** Similar to Twilio, sometimes cheaper for certain regions

Each guide will show you how to integrate with Twilio as the example, but the principles apply to other providers.

# Ready to Get Started?

Click on any of the tool cards above to see detailed step-by-step instructions for implementing the email-to-SMS workflow.

[Zapier Tutorial](#) → [n8n Tutorial](#) → [Node-RED Tutorial](#) → [Vertex AI Tutorial](#) →

Workflow Automation Comparison Guide | Created for developers with limited f

All code examples prioritize maintainability and reliability over performan

Published

Assign To

Edit

### Student Pairs - assignment a3.4

|       |
|-------|
| 1, 10 |
| 11, 9 |
| 12, 8 |
| 13, 7 |
| 14, 6 |
| 15, 5 |
| 16, 4 |
| 2, 3  |

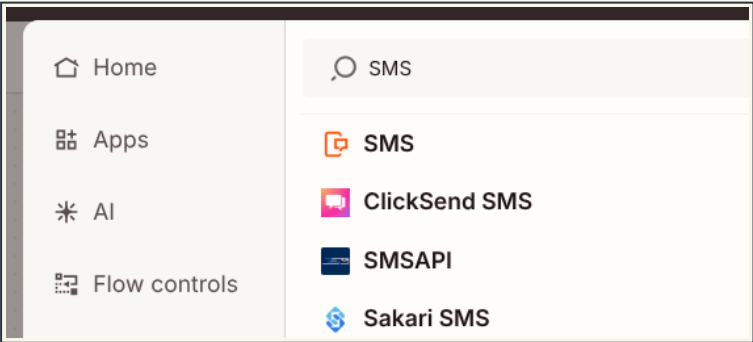
## Zapier: Email to SMS Workflow

The fastest way to set up email-to-SMS notifications with zero coding required

**Update 2025-11-11 SMS instructions**

The Twillio SMS tool is a total pain to use. There are 2 alternatives:

**Option 1:** use the Zapier SMS tool instead. You can find it by adding an action and searching for **SMS**. You will get the list below



Select the top option. Under the hood, this is actually the Twillio action, but you can only send to a number you own. Luckily, the configuration option:

3. Select the event

Setup > Test

App \*

SMS by Zapier

Change

Action event \*

Choose an event

Account \*

Connect SMS by Zapier

Sign in

**Option 2:** use the **PushAlert** **push notification** tool. When you configure the tool, it will ask for a REST API key. use:

**ae04f76a5fe33ed4c54a5917aa8e76df**

The notifications will appear on this page:

<https://mrmxf.com/learn/workflow/assignment-monitor>

## What You'll Build

A "Zap" (Zapier's term for workflow) that monitors your email inbox and sends SMS notifications when emails matching customer inquiries, or important notifications.

**Time Required:** 5-10 minutes | **Difficulty:** Beginner | **Cost:** Free (100 tasks/month)

### ✓ Advantages of Zapier

- Absolutely no coding required
- Fastest setup of all options
- Excellent visual interface
- Built-in email and SMS integrations
- Zero infrastructure management
- Reliable and well-maintained

### ✗ Limitations

- Free tier limited to 100 tasks per month

- Less customization than code-based solutions
- Paid plans can be expensive at scale
- Limited to available app integrations
- Multi-step zaps require paid plan

## Prerequisites

- A Zapier account (sign up free at [zapier.com](https://zapier.com) )
- An email account (Gmail, Outlook, or any IMAP email)
- A Twilio account for SMS (free trial available at [twilio.com](https://twilio.com) )
- Your Twilio Account SID and Auth Token (from Twilio console)
- A Twilio phone number (provided in free trial)

## Step-by-Step Guide

### 1 Create a New Zap

1. Log into your Zapier account at [zapier.com](https://zapier.com) 
2. Click the **"Create Zap"** button in the top left
3. You'll see a blank workflow canvas with two steps: Trigger and Action

**Note:** Zapier calls workflows "Zaps". Each Zap consists of a Trigger (what starts the workflow) and one or more Actions.

### 2 Set Up Email Trigger

1. Click on the **Trigger** step
2. Search for your email provider (e.g., "Gmail", "Outlook", or "Email" for IMAP)
3. Select the email app from the results
4. Choose the trigger event:
  - **"New Email"** - triggers on any new email
  - **"New Email Matching Search"** - triggers only for emails matching criteria (recommended)
5. Click **"Continue"**
6. Connect your email account when prompted (you'll need to authorize Zapier)

**Tip:** Use "New Email Matching Search" to avoid wasting tasks on unimportant emails. You can filter by sender, subject, or body.

### 3 Configure Email Filter (Optional but Recommended)

If you selected "New Email Matching Search":

1. In the **"Search String"** field, enter your filter criteria
2. Examples:
  - `from:important@client.com` - emails from specific sender
  - `subject:urgent` - emails with "urgent" in subject
  - `label:important` - emails with specific label (Gmail)
  - `is:unread` - only unread emails
3. You can combine multiple criteria with spaces
4. Click **"Continue"**
5. Click **"Test trigger"** to verify Zapier can find emails

**Gmail Users:** Use Gmail's search syntax. You can test your filters in Gmail's search box first to ensure they work.

## 4 Add SMS Action

1. Click the **"+"** button below the trigger to add an action
2. Search for "Twilio" or "SMS" in the action search
3. Select **"Twilio"** from the results
4. Choose the action event: **"Send SMS"**
5. Click **"Continue"**
6. Click **"Sign in to Twilio"**
7. Enter your Twilio credentials:
  - **Account SID:** Found in your Twilio console dashboard
  - **Auth Token:** Found in your Twilio console dashboard (click to reveal)
8. Click **"Yes, Continue"**

## 5 Configure SMS Message

1. In the **"From Number"** field, select your Twilio phone number from the dropdown
2. In the **"To Number"** field, enter your mobile phone number (include country code, e.g., +1234567890)
3. In the **"Body"** field, compose your SMS message using email data:
  - Click in the Body field to see available data fields
  - Insert dynamic content from the email (subject, sender, body preview, etc.)

**Example SMS Message:**

New Email Alert!

From: [Insert "From Name" field]

Subject: [Insert "Subject" field]

[Insert first 100 characters of "Body Plain" field]

Sent at [Insert "Date" field]

To insert fields, click where you want the data and select from the dropdown that appears.

**SMS Length Warning:** SMS messages are limited to 160 characters (or 70 for Unicode). Messages longer than this will be truncated, which may affect readability and increase costs. Keep messages concise!

## 6 Test Your Zap

1. Click **"Test action"** at the bottom of the action configuration
2. Zapier will send a test SMS to your phone
3. Check your phone to verify you received the SMS
4. If successful, you'll see a green checkmark
5. If it fails, check:
  - Your phone number format (must include country code, no spaces or dashes)
  - Your Twilio credentials are correct
  - Your Twilio account has credit (free trial includes credit)

## 7 Publish Your Zap

1. Give your Zap a descriptive name (click "Untitled Zap" at the top)
2. Example names: "Urgent Email to SMS", "Client Email Alerts", "VIP Email Notifier"
3. Click the **"Publish"** button in the top right
4. Your Zap is now live and will check for new emails every 15 minutes (on free plan) or every 5 minutes (on paid plan)

**Free Plan Polling:** Zapier checks for new emails every 15 minutes on the free plan. For instant notifications, you need a paid plan with a shorter polling interval.

# Testing Your Workflow

## Verify End-to-End

1. Send yourself a test email that matches your filter criteria
2. Wait up to 15 minutes for Zapier to check your inbox
3. You should receive an SMS notification on your phone

4. Check your Zap's task history:

- Click on your Zap name
- Click the **"Task History"** tab
- You'll see a log of all triggered tasks

## Troubleshooting

### Common Issues and Solutions

Issue: Zap isn't triggering

- **Check filter criteria:** Make sure test emails match your search criteria
- **Wait for polling interval:** Free accounts check every 15 minutes
- **Verify Zap is turned ON:** Check the toggle switch in your Zap list
- **Check task history:** Look for errors in the task log

Issue: SMS not delivering

- **Verify phone number format:** Must be+[country code][number] with no spaces or dashes
- **Check Twilio account:** Ensure you have credit and the number is verified
- **Twilio trial limitations:** Free trials can only send to verified numbers
- **Message too long:** Keep SMS under 160 characters

Issue: Ran out of free tasks

- **Refine filters:** Make email criteria more specific to reduce task usage
- **Upgrade account:** Paid plans start at \$19.99/month for 750 tasks
- **Consider alternatives:** n8n or Node-RED for unlimited free tasks

## Advanced Customization

### Add Multiple Recipients

To send SMS to multiple phone numbers:

1. Duplicate the SMS action (hover over it and click "+")
2. Configure each with a different "To Number"
3. Each recipient will count as a separate task

### Add Filtering Logic

On paid plans, you can add a "Filter" step:

1. Click "+" between trigger and action
2. Select "Filter"

3. Set conditions (e.g., "Only continue if subject contains 'URGENT'")
4. This prevents unnecessary SMS sends and saves tasks

**Free Plan Limitation:** Multi-step Zaps require a paid plan. On the free plan, use email filtering in the trigger instead

## Schedule Quiet Hours

To avoid SMS notifications at night:

1. Add a "Schedule" step (paid plans only)
2. Set active hours (e.g., 8 AM - 10 PM)
3. Or use "Filter by Zapier" to check the time and only continue during business hours

## Cost Considerations

### Understanding Zapier Task Usage

Every time your Zap runs, it uses one task. With the free plan's 100 tasks/month:

- **~3 emails/day** average would use approximately 90 tasks/month
- **Multiple recipients?** Each SMS sent counts as a separate task
- **Failed runs still count** as tasks, so test thoroughly

### Twilio Costs (separate from Zapier)

- **Free trial:** \$15 credit included
- **SMS cost:** Approximately \$0.0075 per SMS sent (varies by country)
- **100 SMS per month:** ~\$0.75/month after trial

## Next Steps

- Monitor your Zap's task usage in the Zapier dashboard
- Refine email filters to reduce unnecessary triggers
- Consider upgrading to a paid plan if you exceed 100 tasks/month
- Explore other Zapier integrations (Slack, database updates, etc.)
- Set up email alerts if your Zap fails

**Pro Tip:** Create a dedicated email label or folder for "urgent" emails, then filter your Zap to only watch that label. This gives you notifications without complex filter rules.

← [Back to Tool Comparison](#)

 Published

 Assign To

 Edit



### Student Pairs - assignment a3.4

|       |
|-------|
| 1, 10 |
| 11, 9 |
| 12, 8 |
| 13, 7 |
| 14, 6 |
| 15, 5 |
| 16, 4 |
| 2, 3  |

## n8n: Email to SMS Workflow

Free unlimited automation with a visual workflow editor

### What You'll Build

An n8n workflow that monitors your email and sends SMS notifications. Unlike Zapier, n8n is free for unlimited workflows for those who don't mind a bit of setup.

**Time Required:** 15-30 minutes | **Difficulty:** Intermediate | **Cost:** Free (self-hosted)

#### ✓ Advantages of n8n

- Unlimited workflows and executions (free)
- Visual workflow builder (like Zapier)
- Self-hosted means full data control

- Can add custom JavaScript logic
- Active community and good documentation
- No task limits or execution caps

#### ✗ Limitations

- Requires self-hosting (or cloud subscription)
- Need basic server/Docker knowledge
- More setup time than Zapier
- You manage updates and maintenance
- Cloud version has costs similar to Zapier

## Installation Options

### Option 1: n8n Cloud Trial (Easiest)

Best for testing before committing to self-hosting.

1. Go to [n8n.io](https://n8n.io) 
2. Click "Start free" for a cloud trial
3. Sign up with email
4. You'll have access to the n8n editor immediately
5. Skip to "Building the Workflow" section below

**Note:** Cloud trial is free but limited. For production use, self-hosting is recommended for unlimited free workflows.

### Option 2: Docker Installation (Recommended for Self-Hosting)

Requires Docker on your computer or server.

```
# Pull the n8n Docker image
docker pull n8nio/n8n

# Run n8n with data persistence
docker run -it --rm \
  --name n8n \
  -p 5678:5678 \
  -v ~/.n8n:/home/node/.n8n \
  n8nio/n8n

# Access n8n at http://localhost:5678
```

#### What this does:

- `-p 5678:5678` - makes n8n accessible on port 5678
- `-v ~/.n8n:/home/node/.n8n` - persists your workflows

- `--rm` - removes container on stop (workflows are saved in volume)

## Option 3: npm Installation

For developers who prefer npm.

```
# Install n8n globally
npm install -g n8n

# Start n8n
n8n start

# Access at http://localhost:5678
```

## Prerequisites

- n8n installed (via cloud, Docker, or npm)
- Email account (IMAP access enabled)
- Twilio account with Account SID, Auth Token, and phone number
- Your phone number for SMS notifications

## Building the Workflow

### 1 Create New Workflow

1. Open n8n in your browser (<http://localhost:5678> or your cloud URL)
2. Click the "+" button or **"New Workflow"**
3. You'll see a blank canvas with a "Start" node
4. Give your workflow a name: click "My workflow" at the top and rename to "Email to SMS Alert"

### 2 Add Email Trigger Node

1. Click the "+" button next to the Start node
2. In the search, type "IMAP Email"
3. Select **"IMAP Email"** from the results
4. Configure the node:
  - **Trigger on:** "New Email"
  - **Host:** Your email server
    - Gmail: `imap.gmail.com`

- Outlook: `outlook.office365.com`

- **Port:** 993 (SSL/TLS)
- **User:** Your email address
- **Password:** Your email password or app-specific password

5. Click "Execute Node" to test the connection

**Gmail Users:** You need to enable "Less secure app access" or use an App Password:

1. Go to Google Account settings
2. Security → 2-Step Verification → App passwords
3. Generate an app password for "Mail"
4. Use this password in n8n

### 3 Add Email Filter (Optional)

To only process certain emails:

1. Click "+" after the IMAP node
2. Search for "IF" node
3. Select "IF"
4. Configure conditions:
  - Click "Add Condition"
  - **Value 1:** Click and select `{{ $json["from"]["value"][0]["address"] }}`
  - **Operation:** "Contains"
  - **Value 2:** Type the sender email or domain you care about
5. Connect the "true" output to the next node (SMS)

**Filter Examples:**

- Subject contains "urgent": `{{ $json["subject"] }}` contains "urgent"
- From specific sender: `{{ $json["from"]["value"][0]["address"] }}` equals "boss@company.com"
- Has attachments: `{{ $json["attachments"].length }}` > 0

### 4 Add Twilio SMS Node

1. Click "+" after your IMAP node (or IF node if using filter)
2. Search for "Twilio"
3. Select "Twilio"
4. Click "Create New Credential"

5. Enter your Twilio credentials:

- **Account SID:** From Twilio dashboard
- **Auth Token:** From Twilio dashboard

6. Click "Create"

## 5 Configure SMS Message

1. In the Twilio node settings:

- **Resource:** SMS
- **Operation:** Send
- **From:** Your Twilio phone number (+1234567890)
- **To:** Your personal phone number (+1234567890)
- **Message:** Click to compose using email data

**Building the Message:** Click in the Message field and select from available data:

```
New Email Alert!

From: {{ $json["from"]["value"][0]["name"] }}
Email: {{ $json["from"]["value"][0]["address"] }}
Subject: {{ $json["subject"] }}

{{ $json["textPlain"].substring(0, 100) }}...

Time: {{ $json["date"] }}
```

**How to insert variables:**

- Type `{{` and a dropdown will appear
- Select the data field you want
- Use `.substring(0, 100)` to limit text length

### 6 Test the Workflow

1. Send yourself a test email
2. In n8n, click **"Execute Workflow"** button at the bottom
3. Wait a moment - n8n will check for new emails
4. Check your phone for the SMS
5. In n8n, you'll see green checkmarks if successful
6. Click on any node to see the data that passed through it

**Debugging Tip:** Click on each node to view the JSON data flowing through your workflow. This helps you understand what is happening.

### 7 Activate the Workflow

1. Click the toggle switch at the top right to **"Active"**
2. Your workflow is now live and will check for emails based on the IMAP trigger settings
3. By default, n8n checks every minute for new emails

**Polling Interval:** The IMAP trigger checks for new emails every 60 seconds by default. You can adjust this in the trigger node settings.

### Advanced Customizations

#### Extract Specific Email Content

Add a "Function" node to process email content with JavaScript:

```
// Function node to extract and format email data
const from = $input.item.json.from.value[0];
const subject = $input.item.json.subject;
const body = $input.item.json.textPlain;

// Extract first 150 characters of body
const preview = body.substring(0, 150).replace(/\n/g, ' ');

// Format SMS message
const smsMessage = `Email from ${from.name}
Subject: ${subject}
${preview}...`;

return {
  json: {
    smsMessage: smsMessage,
    fromName: from.name,
    fromEmail: from.address,
    subject: subject
  }
};
```

Then use `{{ $json["smsMessage"] }}` in your Twilio node.

## Add Time-Based Filtering

Only send SMS during business hours:

1. Add a "Function" node after the IMAP trigger
2. Add this code:

```
// Only continue during business hours (9 AM - 6 PM)
const now = new Date();
const hour = now.getHours();

if (hour >= 9 && hour < 18) {
  return $input.item;
} else {
  // Return empty to stop workflow
  return null;
}
```

## Multiple SMS Recipients

Send to multiple people:

1. Add a "Split In Batches" node
2. Add a "Function" node with recipient list:

```
// List of recipients
const recipients = [
  { name: "John", phone: "+1234567890" },
  { name: "Jane", phone: "+0987654321" }
];

return recipients.map(recipient => ({
  json: {
    ...$ input.item.json,
    recipientPhone: recipient.phone,
    recipientName: recipient.name
  }
}));
```

```
}  
}});
```

Then use `{{ $json["recipientPhone"] }}` in the Twilio "To" field.

## Add Error Handling

Get notified if the workflow fails:

1. Click "Workflow" menu → "Settings"
2. Enable "Error Workflow"
3. Create a separate workflow that sends you an email on errors

## Troubleshooting

### Common Issues

#### IMAP Connection Failed

- **Gmail:** Enable IMAP in Gmail settings, use app password
- **Two-factor auth:** Must use app-specific password
- **Wrong host/port:** Verify your email provider's IMAP settings
- **Firewall:** Ensure port 993 is open

#### Workflow Not Triggering

- **Workflow inactive:** Check the toggle switch is "Active"
- **No new emails:** IMAP only triggers on NEW emails after activation
- **Check execution log:** Click "Executions" in sidebar to see logs

#### SMS Not Sending

- **Twilio credentials:** Verify SID and token are correct
- **Phone format:** Must be +[country code][number]
- **Twilio trial:** Can only send to verified numbers
- **Check node output:** Click Twilio node to see error message

## Production Deployment

### Running n8n 24/7

For reliable operation:

Option 1: Docker with restart policy

```
docker run -d \
  --name n8n \
  --restart unless-stopped \
  -p 5678:5678 \
  -v ~/.n8n:/home/node/.n8n \
  n8nio/n8n
```

#### Option 2: Docker Compose

```
version: "3.7"

services:
  n8n:
    image: n8nio/n8n
    restart: unless-stopped
    ports:
      - "5678:5678"
    environment:
      - N8N_BASIC_AUTH_ACTIVE=true
      - N8N_BASIC_AUTH_USER=admin
      - N8N_BASIC_AUTH_PASSWORD=yourpassword
    volumes:
      - ~/.n8n:/home/node/.n8n
```

#### Option 3: Cloud VPS

Deploy on DigitalOcean, AWS, or similar:

- Create a small instance (\$5-10/month)
- Install Docker
- Run n8n with restart policy
- Set up SSL with Caddy or nginx

## Cost Analysis

#### n8n Costs

- **Self-hosted (Docker/npm):** FREE - unlimited workflows and executions
- **Infrastructure:** \$0 (local) to \$5-10/month (VPS)
- **Cloud version:** Starts at \$20/month for 5,000 executions

#### Twilio Costs (same as Zapier)

- ~\$0.0075 per SMS sent
- 100 SMS/month ≈ \$0.75/month

#### Total Monthly Cost

- **Self-hosted locally:** ~\$0.75 (just SMS)
- **Self-hosted on VPS:** ~\$6-11 (VPS + SMS)
- **n8n Cloud:** ~\$21 (platform + SMS)

## Next Steps

- Explore n8n's 350+ integrations
- Join the n8n community forum for support

- Learn JavaScript to unlock custom node functionality
- Set up workflow backups (export as JSON)
- Configure webhook triggers for instant notifications

**Pro Tip:** n8n workflows can be exported as JSON and version controlled with git. This makes them easy to backup, share, and deploy across environments. Click "Workflow" → "Download" to export your workflow.

[← Back to Tool Comparison](#)

n8n Workflow Guide | Free unlimited automation with full control

**Points** 0

**Submitting** a text entry box or a website url

| Due | For      | Available from | Until |
|-----|----------|----------------|-------|
| -   | Everyone | -              | -     |

 [Rubric](#)

Published

Assign To

Edit

### Student Pairs - assignment a3.4

|       |
|-------|
| 1, 10 |
| 11, 9 |
| 12, 8 |
| 13, 7 |
| 14, 6 |
| 15, 5 |
| 16, 4 |
| 2, 3  |

## Node-RED: Email to SMS Workflow

Try instantly in browser playgrounds OR deploy with Docker - full control for developers

### What You'll Build

A Node-RED flow that monitors email via IMAP and sends SMS notifications through Twilio. Node-RED offers unmatched ease of use and flow-based programming.

**Time Required:** 5-45 minutes (depending on deployment option) | **Difficulty:** Beginner to Advanced | **Cost:** Free (open source)

🌟 **New!** Try Node-RED instantly in online playgrounds before installing locally!

#### ✓ Advantages of Node-RED

- Complete control and customization

- Extensive npm package ecosystem
- Try online first - no installation needed
- Great for IoT and complex integrations
- Active community with many contrib nodes
- Built-in debug tools
- Can integrate with any API or service

#### ✗ Limitations

- Online playgrounds have time/feature limits
- Production use requires self-hosting
- More setup for permanent deployment
- JavaScript knowledge helpful
- Need to manage updates (self-hosted)

## Prerequisites

#### For Online Playgrounds (Quick Start):

- Web browser (Chrome, Firefox, Safari, Edge)
- No installation required!
- Optional: Twilio account for SMS testing

#### For Local/Production Installation:

- Docker installed on your system
- Basic terminal/command line knowledge
- IMAP-enabled email account
- Twilio account (Account SID, Auth Token, phone number)
- Your phone number for notifications

## Getting Started Options

#### Choose Your Path:

- 🚀 **Quick Test (5-10 minutes):** Try online playgrounds - no installation!
- 💻 **Learning/Development (20-30 minutes):** Docker installation on your computer
- 🏢 **Production (30-45 minutes):** Docker with persistence and security

### 1A Option A: Online Playgrounds (Start Here!)

Test Node-RED immediately in your browser with no installation. Perfect for learning the interface and building your first

🚀 Siemens MindConnect Playground (Recommended for Learning)

URL: [playground.mindconnect.rocks](https://playground.mindconnect.rocks) ➞

**Best for:** Quick testing, learning Node-RED, prototyping flows

1. Visit [playground.mindconnect.rocks](https://playground.mindconnect.rocks) 
2. Click **"Start"** or **"Launch Playground"**
3. You'll get a temporary Node-RED instance (lasts 60 minutes)
4. Full Node-RED editor with most standard nodes
5. Can install additional nodes via palette manager
6. Export your flows as JSON before session expires

**MindConnect Playground Features:**

-  No signup required - instant access
-  60-minute sessions (can restart)
-  Full Node-RED editor access
-  Install community nodes from palette
-  Export/import flows as JSON
-  No persistent storage (save your work!)
-  Limited external network access (some APIs blocked)

 **CodeSandbox Node-RED** (Best for Sharing & Collaboration)

**URL:** [codesandbox.io](https://codesandbox.io) 

**Best for:** Sharing flows with others, collaboration, version control

1. Go to [codesandbox.io](https://codesandbox.io) 
2. Sign up/login (free account available)
3. Click **"Create Sandbox"**
4. Search for "node-red" in community templates or search online for shared Node-RED sandboxes
5. Alternative: Create a new Node.js sandbox and add Node-RED
  - Select "Node" or "Vanilla" template
  - In terminal: `npm install -g node-red && node-red`
  - Open preview in new tab
6. Your sandbox persists and can be shared via URL

**CodeSandbox Advantages:**

-  Persistent storage (saved to your account)
-  Shareable URLs - great for getting help
-  Fork and modify existing sandboxes
-  Git integration available
-  Collaboration features (multiple cursors)
-  Requires free account
-  Limited to container resources
-  May need Pro for always-on instances

 **Using Online Playgrounds for This Tutorial**

You can follow along with most of this tutorial using either playground:

1. **Build the flow** in the playground editor (skip Docker steps)
2. **Test with mock data** instead of real email:
  - Use **Inject** nodes to simulate emails
  - Use **Function** nodes to create test email data
  - Use **Debug** nodes to see SMS output instead of sending real SMS
3. **Export your flow** (Menu → Export → Clipboard)
4. **Save the JSON** to your computer (copy to text file)
5. When ready for production, **import to your own Node-RED instance**

#### Playground Limitations for This Workflow:

- ⚠ IMAP email connections may be restricted (firewall/security)
- ⚠ External API calls (Twilio) might have rate limits or be blocked
- ⚠ Sessions expire - export and save your work frequently!
- ⚠ Not suitable for 24/7 production use

**Recommendation:** Use playgrounds to learn and prototype, then deploy to Docker for production.

💡 Playground Workaround: Mock the Workflow

To test the complete flow logic in a playground without real email/SMS:

```
// Use an Inject node with this JSON to simulate an email:
{
  "from": "test@example.com",
  "topic": "Urgent: Server Down",
  "payload": "The production server is experiencing issues...",
  "date": new Date()
}

// Then use Debug nodes instead of Twilio to see SMS output
// This lets you perfect the logic before deploying
```

## 1B Option B: Docker Setup (For Production Use)

Create a directory for Node-RED and set up Docker Compose:

```
# Create directory for Node-RED
mkdir -p ~/nodered
cd ~/nodered

# Create docker-compose.yml
cat > docker-compose.yml << 'EOF'
version: "3.7"

services:
  nodered:
    image: nodered/node-red:latest
    container_name: nodered
    restart: unless-stopped
    ports:
      - "1880:1880"
    volumes:
      - ./data:/data
    environment:
      - TZ=America/New_York
```

E0F

```
# Start Node-RED
docker-compose up -d
```

**What this does:**

- Creates a persistent data directory for your flows
- Maps port 1880 for web access
- Sets restart policy for reliability
- Configures timezone (change as needed)

## 2 Access Node-RED

**For Playgrounds:** Already open in your browser!

**For Docker:**

1. Open your browser to `http://localhost:1880`
2. You'll see the Node-RED editor interface
3. The left panel shows available nodes
4. The center is your flow canvas
5. The right panel shows info and debug output

## 3 Install Required Nodes

Node-RED needs additional nodes for IMAP and Twilio. This works in both playgrounds and local installations:

1. Click the hamburger menu (top right) → **Manage palette**
2. Click the **"Install"** tab
3. Search for and install these nodes:
  - `node-red-node-email` - for IMAP support
  - `node-red-contrib-twilio` - for SMS sending
4. Click "Install" for each
5. Wait for installation to complete

**Installation Note:** Node-RED will restart after installing nodes. Your browser will reconnect automatically.

## Building the Flow

**Two Approaches:**

- **Production Flow:** Steps 4-8 below (uses real IMAP email)
- **Playground/Test Flow:** See "Playground Alternative" in Step 4 (uses mock data)

## 4 Add Email Input Node (Production) OR Inject Node (Playground)

 For Production/Docker Installations:

Use the email input node to monitor real email:

1. In the left panel, find the **"email in"** node (under "network")
2. Drag it onto the canvas
3. Double-click the node to configure:
  - **Protocol:** IMAP
  - **Server:** imap.gmail.com (or your provider)
  - **Port:** 993
  - **Use TLS:** ✓ (checked)
  - **Username:** your email address
  - **Password:** your password or app password
  - **Folder:** INBOX
  - **Disposition:** Read (marks email as read after processing)
  - **Repeat:** 60 (check every 60 seconds)
4. Click **"Done"**

 For Online Playgrounds (Alternative Approach):

Use an inject node to simulate email arrivals - this lets you test the flow logic without needing real email access:

1. In the left panel, find the **"inject"** node (under "common")
2. Drag it onto the canvas
3. Double-click the node to configure:

```
// Set msg.payload to:
{
  "from": "boss@company.com",
  "topic": "Urgent: Production Issue",
  "payload": "The production server is experiencing high load and needs immediate attention. C
out.",
  "date": new Date().toISOString(),
  "attachments": []
}

// Change "Inject once after" to blank
// Set "Repeat" to "none"
// Click "Done"
```

**Playground Testing Tip:** Create multiple inject nodes with different mock emails to test various scenarios:

- Urgent emails (should trigger SMS)
- Regular emails (should be filtered out)
- Emails from specific senders
- Emails with attachments

Click each inject node's button to "send" that email through your flow!

**Gmail App Passwords:** If using Gmail with 2FA:

1. Go to Google Account → Security
2. 2-Step Verification → App passwords
3. Create app password for "Mail"
4. Use this 16-character password in Node-RED

## 5 Add Filter Function (Optional)

Filter emails before sending SMS:

1. Drag a **"function"** node onto the canvas
2. Connect the email node output to the function node input
3. Double-click the function node
4. Name it "Filter Important Emails"
5. Enter this JavaScript code:

```
// Filter logic - only process important emails
// This reduces unnecessary SMS sends

// Example: Only emails from specific sender
if (msg.from && msg.from.includes("important@client.com")) {
  return msg;
}

// Example: Only emails with "urgent" in subject
if (msg.topic && msg.topic.toLowerCase().includes("urgent")) {
  return msg;
}

// Example: Only emails with attachments
if (msg.attachments && msg.attachments.length > 0) {
  return msg;
}

// If no conditions match, return null (stops flow)
return null;
```

**Message Structure:** The email node provides:

- `msg.topic` - email subject
- `msg.from` - sender address
- `msg.payload` - email body (HTML or text)
- `msg.date` - received date
- `msg.attachments` - array of attachments

## 6 Format SMS Message

Create the SMS content:

1. Add another **"function"** node
2. Connect it after the filter (or directly after email if no filter)
3. Name it "Format SMS"
4. Enter this code:

```
// Extract email data
const from = msg.from || "Unknown";
const subject = msg.topic || "No Subject";
const body = msg.payload || "";

// Extract first 100 characters of body text
// Strip HTML if present
let bodyText = body;
if (typeof body === 'string') {
  // Remove HTML tags
  bodyText = body.replace(/<[^>]*>/g, '');
  // Remove extra whitespace
  bodyText = bodyText.replace(/\s+/g, ' ').trim();
  // Limit to 100 characters
  bodyText = bodyText.substring(0, 100);
}

// Format SMS message (keep under 160 chars for single SMS)
const smsText = `Email Alert!
From: ${from}
Subj: ${subject}
${bodyText}...`;

// Prepare message for Twilio
msg.payload = smsText;

// Store original data for debugging
msg.originalEmail = {
  from: from,
  subject: subject,
  date: msg.date
};

return msg;
```

**SMS Length:** Keep messages under 160 characters for a single SMS. Longer messages will be split and cost more messages.

## 7 Add Twilio SMS Node (Production) OR Debug Node (Playground)

 For Production - Real SMS via Twilio:

1. Find **"twilio out"** in the left panel
2. Drag it onto the canvas
3. Connect the format function output to this node
4. Double-click to configure:
  - **Account SID:** From Twilio console

- **Auth Token:** From Twilio console
- **From:** Your Twilio phone number (+12345678901)
- **To:** Your personal phone (+12345678901)
- **Name:** "Send SMS Alert"

5. Click **"Done"**

 For Playgrounds - Simulate SMS with Debug:

Skip Twilio and use a debug node to see what SMS would be sent:

1. Find **"debug"** in the left panel (under "common")
2. Drag it onto the canvas
3. Connect the format function output to this debug node
4. Double-click and configure:
  - **Output:** "complete msg object"
  - **Name:** "SMS Output (would send)"
5. Click **"Done"**
6. When you test, check the debug panel (right side) to see your formatted SMS message

**Playground Testing:** Using debug instead of Twilio lets you:

- See exactly what SMS would be sent
- Test without spending Twilio credit
- Verify message formatting is correct
- Debug your flow logic before production
- No need for Twilio credentials

**Twilio Credentials Location (for production):**

1. Log into [console.twilio.com](https://console.twilio.com) 
2. Dashboard shows Account SID
3. Click "Show" to reveal Auth Token
4. Phone number is in Phone Numbers section

8

## Add Additional Debug Output

Essential for troubleshooting (works in both production and playgrounds):

1. Drag another **"debug"** node onto canvas
2. Connect it to your last node (Twilio or previous debug)
3. This will show successful SMS sends in the debug panel
4. Double-click and set "Output" to "complete msg object"

5. Name it "Flow Complete"

## 9 Deploy and Test

 For Production (Docker/Local):

1. Click the red **"Deploy"** button (top right)
2. Your flow is now active
3. Send yourself a test email that matches your criteria
4. Wait up to 60 seconds for IMAP to check
5. Check your phone for SMS
6. Check the debug panel (right sidebar) for messages

 For Playgrounds:

1. Click the red **"Deploy"** button (top right)
2. Open the debug panel (click bug icon in top right)
3. Click the button on your **Inject** node (left side of the node)
4. Watch the flow execute - nodes will flash as data flows through
5. Check the debug panel to see your formatted SMS message
6. Click inject again to test multiple times
7. **Remember:** Export your flow before the session expires!

### Playground Users - Save Your Work!

- Click hamburger menu → Export → Clipboard
- Copy the JSON to a text file on your computer
- Do this every 15-20 minutes or after major changes
- You can import this JSON into any Node-RED instance later

### Debugging:

- Blue dots on nodes = flow needs deployment
- Red triangle = configuration error
- Click the bug icon (top right) to open debug panel
- Add debug nodes after each function to trace data flow

## Advanced Features

### Add Rate Limiting

Prevent SMS spam if many emails arrive:

1. Add a **"delay"** node before Twilio
2. Configure:
  - **Action:** Rate Limit
  - **Rate:** 1 message per 5 minutes
  - **Drop intermediate messages:** ✓

## Add Business Hours Check

```
// Only send SMS during business hours
const now = new Date();
const hour = now.getHours();
const day = now.getDay(); // 0=Sunday, 6=Saturday

// Check if weekday and between 9 AM - 6 PM
if (day >= 1 && day <= 5 && hour >= 9 && hour < 18) {
  return msg;
} else {
  // Optionally store for later or just drop
  node.warn("Outside business hours - SMS not sent");
  return null;
}
```

## Store in Database

Log all email alerts:

1. Install `node-red-node-sqlite` from palette
2. Add sqlite node to store email metadata
3. Useful for audit trail and debugging

## Multiple Recipients

Send to multiple phone numbers:

```
// Create array of recipients
const recipients = [
  "+12345678901",
  "+10987654321",
  "+11234567890"
];

// Create multiple messages
const messages = recipients.map(phone => ({
  payload: msg.payload, // SMS text
  to: phone,
  from: msg.from // Twilio number set in node config
}));

return [messages];
```

Set the function node "Outputs" to the number of recipients.

## Troubleshooting

### Common Issues

#### Email Node Not Receiving

- **Check credentials:** Verify username/password
- **IMAP not enabled:** Enable in email settings
- **Port/server wrong:** Verify provider's IMAP settings
- **Check node status:** Look for error indicators

#### Twilio Errors

- **21211:** Invalid phone number format
- **21408:** Permission denied (check trial restrictions)
- **21606:** Invalid "from" number
- **Check debug panel:** Shows full error messages

#### Function Node Errors

- **Use `node.warn()`** : Output to debug panel
- **Use try-catch:** Prevent crashes
- **Check msg structure:** Use debug nodes

#### Example Error Handling:

```
try {
  // Your code here
  const from = msg.from || "Unknown";
  // ... rest of code
  return msg;
} catch (error) {
  node.error("Error in function: " + error.message);
  return null;
}
```

## Production Deployment

### Secure Your Installation

Add authentication to Node-RED:

1. Stop Node-RED: `docker-compose down`

2. Edit settings file:

```
# Generate password hash
docker run -it nodered/node-red npx node-red admin hash-pw

# Edit data/settings.js
# Add this section:
```

```
adminAuth: {
  type: "credentials",
  users: [{
    username: "admin",
    password: "$2b$08$..." // your hashed password
    permissions: "*"
  }]
}
```

3. Restart: `docker-compose up -d`

## Backup Your Flows

Regular backups are essential:

```
# Flows are stored in:
~/nodered/data/flows.json

# Backup command:
cp ~/nodered/data/flows.json ~/nodered/backup-$(date +%Y%m%d).json

# Automated daily backup:
echo "0 2 * * * cp ~/nodered/data/flows.json ~/nodered/backup-$(date +%Y%m%d).json" | cron
```

## Costs

### Total Monthly Cost

- **Node-RED:** FREE (open source)
- **Infrastructure:**
  - Local Docker: \$0
  - VPS hosting: \$5-10/month
- **Twilio SMS:** ~\$0.0075 per message
- **100 emails/month:** ~\$0.75 in SMS costs

**Total: \$0.75 - \$11/month** depending on hosting choice

## Next Steps

- Explore Node-RED's extensive node library
- Join the Node-RED forum and Slack community
- Learn flow-based programming concepts
- Integrate with other services (databases, APIs, IoT devices)
- Set up HTTPS with reverse proxy (nginx/Caddy)

**Pro Tip:** Node-RED excels at complex logic and transformations. If you find yourself needing to parse email content, external services, Node-RED's function nodes give you full JavaScript power to handle any scenario.

Published

Assign To

Edit

### Student Pairs - assignment a3.4

|       |
|-------|
| 1, 10 |
| 11, 9 |
| 12, 8 |
| 13, 7 |
| 14, 6 |
| 15, 5 |
| 16, 4 |
| 2, 3  |

## Google Vertex AI Agent Builder: Email to SMS

No-code AI agent automation with Google Cloud's enterprise platform

### What You'll Build

A Google Vertex AI Agent that monitors Gmail and sends SMS notifications via Twilio. Vertex AI Agent Builder provides automation agents powered by Google's AI infrastructure.

**Time Required:** 20-35 minutes | **Difficulty:** Intermediate | **Cost:** Free tier available

#### ✓ Advantages

- No-code visual agent builder
- Enterprise-grade Google Cloud infrastructure
- Built-in Gmail integration

- AI-powered natural language understanding
- Scalable and reliable
- Multi-channel support (email, chat, voice)
- Version control and testing built-in

#### X Limitations

- Requires Google Cloud account (credit card for verification)
- More complex setup than Zapier
- Free tier has usage limits
- Learning curve for Google Cloud Console
- May incur costs at scale

## Prerequisites

- Google Cloud account (free tier available at [cloud.google.com/free](https://cloud.google.com/free) )
- Credit card for account verification (won't be charged on free tier)
- Gmail account for email monitoring
- Twilio account (Account SID, Auth Token, phone number)
- Your phone number for SMS notifications

#### Google Cloud Free Tier:

- \$300 credit for first 90 days
- Vertex AI includes free monthly usage
- No automatic charges after trial ends
- Must manually upgrade to paid account

## Step-by-Step Guide

### 1 Set Up Google Cloud Project

1. Go to [console.cloud.google.com](https://console.cloud.google.com) 
2. Sign in with your Google account
3. Click **"Select a project"** → **"New Project"**
4. Name your project: "Email-SMS-Agent"
5. Click **"Create"**
6. Wait for project creation (30 seconds)
7. Ensure your new project is selected in the top bar

**First-time Setup:** If this is your first Google Cloud project, you'll be prompted to:

- Accept terms of service
- Set up billing (free tier - no charges without upgrade)

- Verify your credit card (no charge, just verification)

## 2 Enable Required APIs

1. In Google Cloud Console, open the navigation menu (☰)
2. Go to **"APIs & Services"** → **"Library"**
3. Search for and enable these APIs (click "Enable" for each):
  - **Vertex AI API** - for agent builder
  - **Gmail API** - for email monitoring
  - **Cloud Functions API** - for webhook handling
4. Wait for each API to be enabled (~30 seconds each)

**Quick Enable:** You can enable all APIs at once by searching for each and clicking "Enable" in sequence. The console

## 3 Create Vertex AI Agent

1. In Google Cloud Console navigation (☰), go to **"Vertex AI"**
2. Click **"Agent Builder"** in the left sidebar
3. Click **"Create Agent"**
4. Choose **"Start from scratch"**
5. Configure agent:
  - **Agent name:** "Email-to-SMS-Notifier"
  - **Description:** "Monitors Gmail and sends SMS alerts for important emails"
  - **Language:** English
  - **Time zone:** Your timezone
6. Click **"Create"**

## 4 Configure Gmail Trigger

Set up the agent to monitor Gmail:

1. In your agent, click **"Triggers"** in the left panel
2. Click **"Add Trigger"**
3. Select **"Gmail"** as the trigger type
4. Click **"Connect Gmail Account"**
5. Authorize the Gmail account you want to monitor

6. Configure trigger settings:

- **Trigger on:** New email in inbox
- **Filter:** Add filter criteria (optional but recommended)
  - From specific sender
  - With specific label (e.g., "Important")
  - Subject contains keywords
- **Check frequency:** Every 1 minute

7. Click **"Save Trigger"**

#### Filter Examples:

```
// Only emails from specific sender
from:boss@company.com

// Only emails with "urgent" in subject
subject:urgent

// Only emails with Important label
label:important is:unread

// Combine multiple criteria
from:client@company.com OR subject:urgent
```

## 5 Add Data Extraction Flow

Configure what information to extract from emails:

1. In your agent, click **"Flows"**
2. Click **"Default Start Flow"** to edit
3. Click **"Add Block"** → **"Data Transformation"**
4. Name it: "Extract Email Data"
5. Configure extraction parameters:

```
{
  "from_email": "$trigger.email.from",
  "from_name": "$trigger.email.fromName",
  "subject": "$trigger.email.subject",
  "body_preview": "$trigger.email.bodyPlainText.substring(0, 100)",
  "received_time": "$trigger.email.date",
  "labels": "$trigger.email.labels"
}
```

6. Click **"Save Block"**

**Parameter Reference:** Use `$trigger.email.*` to access email properties. Vertex AI provides autocomplete for

## 6 Add Conditional Logic (Optional)

Filter which emails should trigger SMS:

1. Click **"Add Block"** → **"Condition"**
2. Name it: "Check if SMS Needed"
3. Set condition logic:

```
// Example: Only send SMS for urgent emails
$page.params.subject.contains("urgent") OR
$page.params.subject.contains("ASAP") OR
$page.params.labels.contains("important")
```
4. Click **"Save"**
5. Connect the "true" path to the next step (SMS)
6. Connect the "false" path to an "End Flow" block

## 7 Create SMS Webhook Function

Since Vertex AI doesn't have native Twilio integration, we'll create a Cloud Function webhook:

1. Open a new tab and go to [Cloud Functions](#) 
2. Click **"Create Function"**
3. Configure:
  - **Name:** send-sms-twilio
  - **Region:** Choose nearest region
  - **Trigger:** HTTPS
  - **Authentication:** Allow unauthenticated invocations (for simplicity)
4. Click **"Save"** then **"Next"**
5. Set **Runtime:** Node.js 20
6. Set **Entry point:** sendSMS

Function Code (index.js):

```
/**
 * Cloud Function to send SMS via Twilio
 *
 * This function receives email data from Vertex AI Agent
 * and sends an SMS notification via Twilio API.
 *
 * Design principles:
 * - Clear error handling for reliability
 * - Comprehensive logging for debugging
 * - Validation of required fields
 * - Maintainable code structure
 */

const functions = require('@google-cloud/functions-framework');

// Twilio configuration
// Best practice: Use Secret Manager for production
const TWILIO_ACCOUNT_SID = 'your_account_sid_here';
const TWILIO_AUTH_TOKEN = 'your_auth_token_here';
const TWILIO_FROM_NUMBER = '+12345678901';
const TWILIO_TO_NUMBER = '+12345678901';

/**
```

```

* Main function - sends SMS via Twilio
*
* @param {Object} req - HTTP request object
* @param {Object} res - HTTP response object
*/
functions.http('sendSMS', async (req, res) => {
  // Enable CORS for Vertex AI
  res.set('Access-Control-Allow-Origin', '*');

  if (req.method === 'OPTIONS') {
    res.set('Access-Control-Allow-Methods', 'POST');
    res.set('Access-Control-Allow-Headers', 'Content-Type');
    res.status(204).send('');
    return;
  }

  try {
    // Extract email data from request
    const emailData = req.body;

    // Validate required fields
    if (!emailData || !emailData.subject) {
      throw new Error('Missing required email data');
    }

    // Format SMS message
    const smsMessage = formatSMSMessage(emailData);

    // Send SMS via Twilio
    await sendTwilioSMS(smsMessage);

    // Return success response
    res.status(200).json({
      success: true,
      message: 'SMS sent successfully'
    });

  } catch (error) {
    console.error('Error sending SMS:', error);
    res.status(500).json({
      success: false,
      error: error.message
    });
  }
});

/**
 * Format email data into SMS message
 * Keep under 160 characters for single SMS
 */
function formatSMSMessage(emailData) {
  const from = emailData.from_name || emailData.from_email || 'Unknown';
  const subject = emailData.subject || 'No Subject';
  const preview = emailData.body_preview || '';

  // Truncate to fit SMS length
  const fromTrunc = from.substring(0, 30);
  const subjectTrunc = subject.substring(0, 40);
  const previewTrunc = preview.substring(0, 60);

  return `Email Alert\nFrom: ${fromTrunc}\n${subjectTrunc}\n\n${previewTrunc}`;
}

/**
 * Send SMS via Twilio API
 * Uses basic HTTP request - no Twilio SDK needed
 */
async function sendTwilioSMS(message) {
  const fetch = require('node-fetch');

  const url = `https://api.twilio.com/2010-04-01/Accounts/${TWILIO_ACCOUNT_SID}/Messages.json`;

  // Create Basic Auth header
  const auth = Buffer.from(`${TWILIO_ACCOUNT_SID}:${TWILIO_AUTH_TOKEN}`).toString('base64');

  // Prepare form data
  const params = new URLSearchParams();
  params.append('From', TWILIO_FROM_NUMBER);

```

```

params.append('To', TWILIO_TO_NUMBER);
params.append('Body', message);

const response = await fetch(url, {
  method: 'POST',
  headers: {
    'Authorization': `Basic ${auth}`,
    'Content-Type': 'application/x-www-form-urlencoded'
  },
  body: params
});

if (!response.ok) {
  const errorText = await response.text();
  throw new Error(`Twilio API error: ${errorText}`);
}

return await response.json();
}

```

Package Dependencies (package.json):

```

{
  "name": "send-sms-twilio",
  "version": "1.0.0",
  "dependencies": {
    "@google-cloud/functions-framework": "^3.0.0",
    "node-fetch": "^2.6.7"
  }
}

```

7. Update the Twilio credentials in the code
8. Click **"Deploy"**
9. Wait for deployment (2-3 minutes)
10. Copy the **Trigger URL** from the function details

**Security Note:** For production, use Google Secret Manager to store credentials instead of hardcoding them. This e

## 8 Connect Webhook to Agent

Link the Cloud Function to your Vertex AI Agent:

1. Return to your Vertex AI Agent in the console
2. In the flow editor, click **"Add Block"** → **"Webhook"**
3. Name it: "Send SMS via Twilio"
4. Configure webhook:
  - **URL:** Paste your Cloud Function trigger URL
  - **Method:** POST
  - **Headers:**

```
Content-Type: application/json
```

- **Body:** (Use extracted email data)

```

{
  "from_email": "$session.params.from_email",

```

```
"from_name": "$session.params.from_name",
"subject": "$session.params.subject",
"body_preview": "$session.params.body_preview",
"received_time": "$session.params.received_time"
}
```

5. Click **"Save"**

## 9 Test the Agent

1. In Vertex AI Agent Builder, click **"Test Agent"** (top right)
2. In the test simulator, you can:
  - Manually trigger with test data
  - Use the **"Simulate Trigger"** option
  - Enter sample email data
3. Click **"Test"**
4. Watch the flow execution in the test panel
5. Check your phone for SMS
6. Review logs in the execution history

### Testing Strategy:

1. First: Test Cloud Function directly with Postman or curl
2. Second: Test agent flow with simulator
3. Third: Send real email to trigger end-to-end

## 10 Deploy to Production

1. Review your agent flow one more time
2. Click **"Publish"** in the top right
3. Confirm you want to publish to production
4. Your agent is now live and monitoring Gmail!
5. Send a test email that matches your filters
6. Wait 1-2 minutes (Gmail polling interval)
7. Check your phone for SMS

### Production Checklist:

- ✓ Gmail filters configured correctly
- ✓ Cloud Function deployed successfully
- ✓ Webhook URL is correct
- ✓ Twilio credentials are valid

- ✓ Phone numbers in correct format
- ✓ Tested with simulator

## Monitoring and Debugging

### View Agent Execution History

1. In Vertex AI Agent Builder, click "**History**"
2. See all agent executions with timestamps
3. Click any execution to see detailed flow
4. Review which blocks executed and their outputs
5. Check for errors or warnings

### View Cloud Function Logs

1. Go to [Cloud Functions](#) 
2. Click your `send-sms-twilio` function
3. Click "**Logs**" tab
4. See all function executions
5. Filter by severity (Error, Warning, Info)
6. Click any log entry for details

### Common Issues

#### Agent Not Triggering

- **Gmail authorization:** Re-authorize Gmail connection
- **Filters too strict:** Broaden your email filters
- **Agent not published:** Ensure agent is in production
- **Check history:** See if trigger fired but flow failed

#### SMS Not Sending

- **Cloud Function errors:** Check function logs
- **Twilio credentials:** Verify SID and token
- **Phone format:** Must be +[country][number]
- **Webhook URL:** Ensure correct URL in agent
- **CORS issues:** Check function allows Vertex AI origin

#### Quota Exceeded

- **Gmail API quota:** Default is 1 billion queries/day (should be fine)

- **Vertex AI quota:** Check Quotas page in console
- **Cloud Functions:** 2 million invocations/month free
- **Request quota increase:** Via Google Cloud Console

## Cost Analysis

Google Cloud Costs (Monthly Estimates)

### Vertex AI Agent Builder:

- First 1,000 sessions: FREE
- Each session = one email checked and processed
- \$0.002 per session after free tier
- Example: 10,000 emails/month = \$18

### Gmail API:

- FREE (included with Google Workspace or personal Gmail)
- Generous quota limits

### Cloud Functions:

- First 2 million invocations: FREE
- \$0.40 per million invocations after
- Example: 10,000 SMS/month = FREE

### Twilio SMS:

- ~\$0.0075 per SMS sent
- Example: 100 SMS/month = \$0.75

Total Estimated Cost for Light Usage (100 emails/month):

**\$0.75/month** (just Twilio SMS, everything else in free tier)

Total Estimated Cost for Heavy Usage (10,000 emails/month):

**~\$93.75/month** (\$18 Vertex AI + \$75 Twilio)

## Advanced Features

### AI-Powered Email Classification

Use Vertex AI's natural language capabilities to intelligently classify emails:

1. Add a **"Generative AI"** block to your flow
2. Configure prompt:

```
Analyze this email and determine its urgency level (low/medium/high):  
  
Subject: $session.params.subject  
From: $session.params.from_name  
Preview: $session.params.body_preview
```

```
Return JSON: {"urgency": "low|medium|high", "reason": "brief explanation"}
```

3. Use AI response to decide whether to send SMS
4. Only send SMS for "high" urgency emails

## Multi-Channel Notifications

Send to SMS, Slack, or other channels based on email content:

1. Add multiple webhook blocks
2. Create condition blocks to route based on criteria
3. High urgency → SMS
4. Medium urgency → Slack
5. Low urgency → Log only

## Schedule Quiet Hours

Prevent SMS during nights/weekends:

1. Add a **"Condition"** block
2. Check current time:

```
// Only send SMS during business hours
const now = new Date();
const hour = now.getHours();
const day = now.getDay();

// Weekday (1-5) and between 9 AM - 6 PM
(day >= 1 && day <= 5 && hour >= 9 && hour < 18)
```

3. Queue messages outside hours (use Cloud Tasks)

## Response Tracking

Store notification history in Firestore:

1. Enable Firestore API
2. Add Firestore write to Cloud Function
3. Track: timestamp, email subject, SMS sent status
4. Create dashboard to view notification history

## Security Best Practices

## Use Secret Manager for Credentials

Instead of hardcoding Twilio credentials:

1. Enable Secret Manager API
2. Store credentials:

```
gcloud secrets create twilio-account-sid --data-file=-  
gcloud secrets create twilio-auth-token --data-file=-
```

3. Update Cloud Function to read from Secret Manager
4. Grant function's service account secret access

## Enable Function Authentication

Require authentication for Cloud Function webhook:

1. Edit function → Trigger → Require authentication
2. Create service account for Vertex AI agent
3. Grant invoker role to service account
4. Update webhook in agent to use service account auth

## Rate Limiting

Prevent SMS spam if many emails arrive:

1. Add Redis/Memorystore for rate limit tracking
2. Check if SMS sent in last X minutes
3. If yes, skip or aggregate notifications
4. If no, send SMS and record timestamp

## Comparison with Other Tools

When to Choose Vertex AI Agent Builder:

- ☒ You need AI-powered email classification
- ☒ You want enterprise-grade infrastructure
- ☒ You're already using Google Cloud
- ☒ You need complex multi-step workflows
- ☒ You want built-in testing and versioning
- ☒ You need to scale to high volumes

When to Choose Alternatives:

- Zapier: If you want fastest setup (5 min vs 30 min)
- n8n: If you want self-hosted and free unlimited
- Node-RED: If you're developer wanting full control

## Next Steps

- Explore Vertex AI Agent Builder templates
- Add more sophisticated AI logic
- Integrate with other Google Cloud services
- Set up monitoring dashboards
- Implement A/B testing for different flows
- Create multiple agents for different email types

**Pro Tip:** Vertex AI Agent Builder really shines when you add AI-powered decision making. Use the Generative AI blocks information, or even draft SMS response suggestions based on email content.

[← Back to Tool Comparison](#)

Google Vertex AI Agent Builder Guide | Enterprise AI-powered automations

Points 0

Submitting a text entry box or a website url

| Due | For      | Available from | Until |
|-----|----------|----------------|-------|
| -   | Everyone | -              | -     |

 [Rubric](#)