DevSheets

n8n

Workflow automation with items, expressions and core nodes, in n8n 2.

01 Install & start

```
# n8n 2.36+ needs Node.js 24 or newer
npx n8n              # try it, no install
npm install -g n8n && n8n start
docker volume create n8n_data
docker run -it --rm -p 5678:5678 \
  -v n8n_data:/home/node/.n8n n8nio/n8n
```

Open http://localhost:5678 and create the owner account. npm installs keep their data in ~/.n8n.

02 Workflows, nodes & items

```
// Data between nodes: an array of items
[
  { json: { id: 1, name: 'Alice' } },
  { json: { id: 2, name: 'Bob' } },
]
```

A trigger (Manual, Webhook, Schedule, app event) starts each run; most nodes then run once per item. Edits auto-save as a draft, and Publish puts a version live (it replaced the Active toggle in 2.0).

03 Expressions

```
{{ $json.user.email }}
{{ $json['order-id'] }}
{{ $('HTTP Request').item.json.id }}
{{ $('HTTP Request').first().json.id }}
{{ $('HTTP Request').all().length }}
{{ $workflow.name }}  {{ $execution.id }}
{{ $execution.mode }}  // test or production
```

Wrap expressions in {{ }} in any node field. $json is the current item; $('Node Name').item is the item it came from in an earlier node. $node["Name"].json is the legacy form.

04 Dates with Luxon

```
{{ $now.toISO() }}
{{ $now.toFormat('yyyy-MM-dd HH:mm') }}
{{ $today.plus({ days: 7 }) }}
{{ $now.minus({ hours: 2 }).toRelative() }}
{{ '15/01/2024'.toDateTime('dd/MM/yyyy') }}
```

$now and $today (midnight) are Luxon DateTimes in the workflow time zone. Dates reach nodes as strings, so parse them with .toDateTime(). Tokens: yyyy MM dd HH mm ss, MMMM (January), EEEE (Monday).

05 Transform data

```
{{ $json.email.toLowerCase().trim() }}
{{ $json.tags.join(', ') }}
{{ $json.users.filter(u => u.active) }}
{{ $json.paid ? 'Yes' : 'No' }}
{{ $json.nickname || 'Guest' }}
{{ $jmespath($json, 'users[*].email') }}
```

Plain JavaScript works inside {{ }}. JMESPath is the $jmespath(obj, query) function; filters take backtick literals, as in [?age > `18`].

06 Edit Fields (Set)

```
// Mode: Manual Mapping, Fields to Set
name     {{ $json.first }} {{ $json.last }}
email    {{ $json.email.toLowerCase() }}
user.id  {{ $json.id }}
created  {{ $now.toISO() }}
```

Include Other Input Fields is off by default, so only these fields come out; turn it on to keep the rest. user.id builds a nested object. JSON mode takes an object.

07 If & Switch

`If`	true / false; AND or OR
`Switch: Rules`	One output per rule
`Switch: Expression`	Output Index (a number)
`Fallback Output`	None, Extra Output, Output 0

Conditions are typed (String, Number, Date & Time and more): is equal to, contains, exists, is after. Prefer one Switch to nested IFs.

08 Merge, Aggregate & loops

`Append`	Inputs one after another
`Combine`	Matching Fields, Position, All Possible Combinations
`SQL Query`	SELECT from input1, input2

Aggregate collects items into one list; Split Out does the reverse. Loop Over Items (was Split in Batches) passes Batch Size items per loop.

09 HTTP Request

`Authentication`	Predefined or Generic Credential Type
`Pagination`	Update a Parameter in Each Request, or Response Contains Next URL
`Batching`	Items per Batch, Batch Interval
`Response`	Include Response Headers and Status

Import cURL fills the node from a curl command. Keep secrets in credentials, never in Code nodes or expressions.

DevSheets

n8n

Code, triggers, error handling and self-hosting.

10 Code: all items (default)

```
return $input.all().map(({ json }) => ({
  json: {
    email: json.email.toLowerCase(),
    fullName: `${json.first} ${json.last}`,
  },
}));
```

Mode: Run Once for All Items. Return an array of { json } items; plain objects get wrapped in json for you, but mixing both forms errors.

11 Code: each item

```
const data = $input.item.json; // or $json
data.total = data.price * data.quantity;
const user = $('Get User').item.json;
return { json: { ...data, user } };
```

Mode: Run Once for Each Item; return one object. Code runs on task runners in n8n 2: no HTTP or file access, and $env is blocked unless N8N_BLOCK_ENV_ACCESS_IN_NODE=false.

12 Static data & variables

```
const store =
  $getWorkflowStaticData('global');
const lastId = store.lastId || 0;
const fresh = $input.all()
  .filter((i) => i.json.id > lastId);
store.lastId = Math.max(lastId,
  ...fresh.map((i) => i.json.id));
return fresh;
```

Saved only by published, trigger-started runs, never test runs; 'node' scope is per node. $vars.name reads custom variables (read-only, paid plans).

13 Webhook trigger

```
https://n8n.example.com/webhook-test/orders
https://n8n.example.com/webhook/orders
```

```
{{ $json.body.email }}
{{ $json.query.page }}
{{ $json.headers['x-api-key'] }}
```

The test URL works while listening in the editor, the production URL once published. Auth: Basic, Header or JWT. Max payload 16 MB.

14 Schedule trigger

```
# [second] minute hour day month weekday
*/5 * * * *       # every 5 minutes
0 9 * * 1-5       # weekdays at 9:00
0 9,17 * * *      # 9:00 and 17:00
*/10 * * * * *    # every 10 seconds
```

Or pick a Seconds to Months interval. Uses the workflow time zone, else GENERIC_TIMEZONE. Runs only once published.

15 Error handling

Retry On Fail	Up to 5 tries, 0 to 5000 ms apart
On Error	Stop Workflow, Continue, or Continue (using error output)
Always Output Data	Empty item if nothing returns

```
{{ $json.workflow.name }}
{{ $json.execution.error.message }}
{{ $json.execution.lastNodeExecuted }}
```

An error workflow starts with Error Trigger; pick it in Workflow Settings. It runs for automatic executions only.

16 Production settings

```
N8N_HOST=n8n.example.com
N8N_PROTOCOL=https
N8N_WEBHOOK_URL=https://n8n.example.com/
N8N_ENCRYPTION_KEY=<openssl rand -hex 32>
GENERIC_TIMEZONE=America/New_York
DB_TYPE=postgresdb    # + DB_POSTGRESDB_*
```

Back up the encryption key: credentials can't be decrypted without it. Set TZ to the same zone. Use PostgreSQL in production. Before 2.30: WEBHOOK_URL.

17 Queue mode

```
EXECUTIONS_MODE=queue
QUEUE_BULL_REDIS_HOST=redis
N8N_ENCRYPTION_KEY=<same as main>
docker run -e EXECUTIONS_MODE=queue \
  -e QUEUE_BULL_REDIS_HOST=redis \
  n8nio/n8n worker --concurrency=10
```

Set these on main and every worker. Main takes triggers and webhooks; workers run executions from Redis. Needs PostgreSQL.

18 AI Agent

Chat models	OpenAI, Anthropic, Gemini, Ollama
Memory	Simple, Postgres or Redis Chat
Tools	HTTP Request, Code, Call n8n Workflow
Vector stores	Pinecone, Supabase, Qdrant

The AI Agent node needs a chat model and at least one tool. Chatbots start with a Chat Trigger. Simple Memory doesn't work in queue mode production.