

DRUPALCON ROTTERDAM 2026 · DEVELOPMENT, AI & AGENTIC ARCHITECTURE

SDCs, Canvas, and the Agent That Builds With Them

Matt Glaman · Acquia · Principal Software Engineer

/u/mglaman · @mglaman@phpc.social · /in/mattglaman · #DrupalCon

Drupal Canvas is a **visual page builder**.

Site builders assemble pages and templates from your components.

Single Directory Components, not a new format invented by Drupal Canvas

We built **canvas_tools** so an agent can do what a site builder does in Drupal Canvas.

The session promised **MCP**.

The agent would write SDCs locally, on the filesystem.

It would reach the running site over MCP.

The demo runs on a **CLI**.

Reaching the site never needed MCP.

The agent already had a shell open on it.

A Drush call is a tool call, with no extra steps

Both transports are live · same plugins on both sides

The transport follows the **work**.

Drush runs the tools against a site on your machine.

MCP runs the same tools against one that isn't.

THE WORK

SDCs, theme, config

Where it lives

Files in a working tree.

Committed, reviewed, kept in git.

How the agent reaches it

It just writes them.

No bridge, no API needed.

Agents could always do this half · the gap was the site

THE WORK

Pages, templates, content

Where it lives

State in a running site.

Maybe one you have no shell into.

How the agent reaches it

`local → drush tool:run`

`remote → HTTP /mcp`

This is the half that needed building · one bridge, either transport

You don't pick a transport. **You get both.**

That is what building the bridge at the Tool API layer buys you.

The same plugins answer on the CLI and over MCP, with no extra code.

HOW IT WORKS

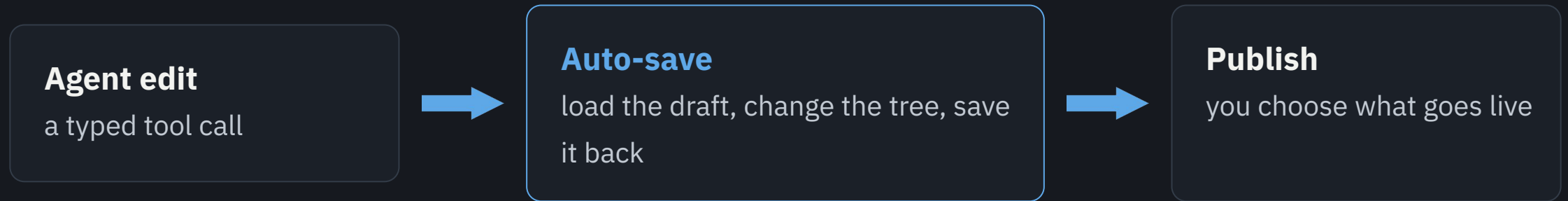
auto-save → publish

Every operation goes through auto-save. Nothing goes live on its own.

Agents get **no special path**.

Every tool call goes through auto-save, the same as the editor.

Agent and editor, same draft



A draft belongs to the page or template, not to whoever is editing it.
So an editor left open sees the agent's changes land.

Auto-save is the integration seam

The stack

canvas

The visual page builder. Component tree, auto-save, publish.

tool

The authoring surface. One typed tool plugin per operation.

tool_belt

Core operations already built: content types, fields, and nodes.

canvas_tools

The bridge. Canvas page building, exposed as tool plugins.

`mcp_server` + `mcp_server_tool_bridge` → the same tools over MCP

Pages, page templates, content templates

page

A standalone landing page at its own path.
Nothing else has to exist for it to render.

page_variant

A theme-independent full-page template.
New in Canvas 1.11.0, replacing theme-global regions.

content_template

How a bundle renders, like [node.article.full](#).
The only one with a host entity to bind against.

All three are component trees, so the same tools drive all three.

One operation, one plugin

```
#[Tool(  
  id: 'canvas_add_component',  
  operation: ToolOperation::Write,  
  destructive: FALSE, // writes to a draft  
  input_definitions: [/* typed inputs */],  
)]  
final class AddComponentTool extends ToolBase {  
  function doExecute($values): ExecutableResult  
  function checkAccess($values, AccountInterface $a): bool  
}
```

An attribute, doExecute(), checkAccess() · the CLI, MCP and the AI module all get it

Only three tools can destroy anything

18

Write to a draft

Discard it, or publish it. Nothing is lost either way.

3

Marked destructive

`delete_page`, `delete_component`, `discard_auto_save`.

Each plugin declares whether it is destructive, so an agent knows before it calls.

That asymmetry is the safety argument

DESIGN + BUILD

design → hand off → build

That is everything the tools can do. Now the workflow a team would actually run.

Design with one agent. **Build with another.**

One produces the design: layout, copy, variants.

The other builds it: SDCs, content model, template, page.

Rotterdam Coffee & Ba...
implies, with a type for each
- the :root design tokens
Use placeholder copy for a Dutch recipe. Stroopwafels is fine.
You answered
ds: Picked a design system (now attached)
lang: English
ingredients: One component per line (IngredientItem, Step) repeated in the page tree
welcome: Hero with shop intro, Opening hours / address, Callout (e.g. seasonal notice), Short about text
handoff_format: Styled HTML doc
images: Image placeholders
Ask user, Designing x11, Listing files, R...
Recipe Page.dc.html
Now the handoff document.
Designing
wrote "recipe content riel..."
Organic
Describe what you want to create...
Opus 5.5 Medium

Welcome Page
11 pages

100%
Comment
Edit
Present
Share

Kade & Kruim

Welcome Recipes

COFFEE & BAKING · ROTTERDAM

Good coffee, and something to go with it.

We roast in small batches and bake every morning in our kitchen on the Nieuwe Binnenweg. This site collects the recipes behind the counter so you can bake them at home.

photo · the counter in the morning

Autumn hours

From 1 October we open at 8:00 on weekdays. Weekend hours stay the same.

The design agent · the welcome page, and a callout in its info variant

Rotterdam Coffee & Ba...

implies, with a type for each

- the :root design tokens

Use placeholder copy for a Dutch recipe. Stroopwafels is fine.

You answered

ds: Picked a design system (now attached)

lang: English

ingredients: One component per line (IngredientItem, Step) repeated in the page tree

welcome: Hero with shop intro, Opening hours / address, Callout (e.g. seasonal notice), Short about text

handoff_format: Styled HTML doc

images: Image placeholders

Ask user, Designing x11, Listing files, R...

Recipe Page.dc.html

Now the handoff document.

Designing

Created Component

Organic

Describe what you want to create...

Opus 5.5 Medium

Recipe Page11 pages

100%

Comment

Edit

Present

Share

M

Kade & Kruim

Welcome

Recipes

RECIPE · DUTCH CLASSICS

Stroopwafels

Two thin waffle wafers baked on a hot iron, split while still warm and sealed with cinnamon caramel syrup. Rest one on top of your coffee cup for a minute and the syrup softens.

photo · stroopwafels stacked on a plate

DIFFICULTY

Medium

PREP

30 min

REST

45 min

BAKE

25 min

MAKES

12 wafels

The same components, laid out for a recipe · this one becomes a content template

Agents work in **components**, not markup.

Add a component, move it, set its props, bind it to a field.

A fixed set of typed operations, nothing to agree on.

What comes out is real: pages, page templates, content templates.

WHAT THE BUILDER AGENT ALREADY KNOWS

"Lay out how a content type renders by building a Canvas content template and **binding component props to the entity's fields.**"

build-content-template/SKILL.md · ships inside canvas_tools

The **skills** ship with the module.

`ai_best_practices` syncs any module's skills into `.agents/skills/` on install.

Anything that reads that folder picks them up.

So `canvas_tools` brings its skills with it.

The agent gets the workflow, not just the tools.

Two kinds of skills

The mechanics

How to call a Drupal tool at all.

Find one, read its contract, run it.

One skill, and it is not about Canvas.

The workflow

How to do one job start to finish.

Which tools, in what order, and how to check it worked.

The rest, one per job.

THE MECHANICS

use-drupal-tools

Find a tool, inspect its contract, then run it.

`tool:search, tool:info, tool:run`

ships in drupal/tool

THE WORKFLOW

author-sdc-component

Write an SDC into the theme, then verify it is Canvas-compatible.

THE WORKFLOW

build-page-template

Build the theme-independent full-page template, and make it the site default.

THE WORKFLOW

build-content-template

Lay out how a bundle renders, and bind component props to the entity's own fields.

THE WORKFLOW

build-canvas-page

Build a standalone landing page, and publish it.

ships in canvas_tools

THE DEMO

author → model → build → publish

The clips ahead are recordings of real runs, not mockups.

01

The agent **authors the SDCs.**

"Build every component the handoff lists as an SDC. Audit them when you are done."

The agent audits its own output

```
# writes callout.component.yml + callout.twig  
drush canvas:component:generate --source=sd  
drush canvas:component:status # not listed = compatible  
drush canvas:component:list | grep callout # [Enabled]
```

A verdict per component, with reasons.

A non-zero exit on incompatible, so the same command gates CI.

What it wrote

```
# callout.component.yml
name: Callout
props:
  type: object
  properties:
    title: { type: string }
    body: { type: string }
    variant: { enum: [info, success, warning] }
```

Typed props and an enum.

The schema is what makes the next two acts possible.

web/themes/custom/rotterdam_qa/components/callout/



Claude Code v2.1.283

Opus 5.5 with high effort · Claude Max

~/Sites/rotterdam-demo

> Try "how does slides.html work?"

→ rotterdam-demo git:(main) ×

▶▶ auto mode on (shift+tab to cycle) · ← 1 agent

02

The agent **models the content.**

"Create the Recipe content type with the fields in the handoff, then the Stroopwafels sample recipe."

A content model, and content in it

```
entity_bundle_add {node, recipe, "Recipe"}
field_storage_add {node, field_difficulty, list_string}
field_add         {node, recipe, field_difficulty, required}
# fourteen fields, two operations each
entity_create     {node, recipe,
  base_fields: {title: "Stroopwafels"},
  fields: {...the rest}} → node <nid>
```

All tool_belt: · a field is two operations, storage then instance



Claude Code v2.1.283

Opus 5.5 with high effort · Claude Max

~/Sites/rotterdam-demo

› Read design/recipe_site/Handoff.dc.html. Create the Recipe content type with the fields in its "Recipe content fields" table. Skip the repeating ingredients and steps rows. Then create the Stroopwafels sample recipe using the examples in that table. Tell me the node id.█

→ rotterdam-demo git:(main) ×

▶▶ auto mode on (shift+tab to cycle)

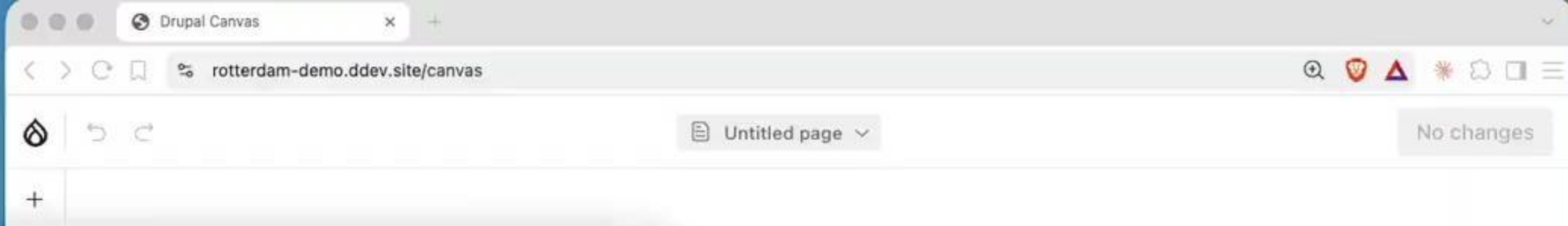
03

The agent **builds the display.**

"Follow the recipe page tree in the handoff. Bind each prop it marks with an arrow."

Bind what Canvas allows. Set the rest.

```
create_content_template {node, recipe, full} → node.recipe.full
add_component           {content_template, sdc...hero} → uuid # ×3
bind_component_props    {uuid, bindings: {
  title: title, intro: field_intro, imageSrc: field_image}}
update_component_props  {uuid, {level: "medium"}} # enum, so static
publish_auto_saves      {content_template, ...} → live
```



The repeating ingredients and steps rows were skipped, as you asked.

* Cooked for 3m 8s · done 5:54 AM

* recap: I set up the Recipe content type from the handoff and created the Stroopwafels sample recipe, which is node 1. Next, replace the two cut-off example texts and decide how to handle the slug URL. (disable recaps in /config)

Create a Canvas content template for the Recipe full view, then stop so I can open the editor.

● Skill(build-content-template)
└ Successfully loaded skill

* Canoodling... (2s · ↓ 169 tokens · thinking with high effort)
└ Tip: Continue your session in Claude Code Desktop with /desktop

→ rotterdam-demo git:(main) ×

▶▶ auto mode on (shift+tab to cycle) · ← 1 agent

me to Drupal Canvas! This tool allows you to and manage your content experiences with you can add components, customize layouts, view your changes in real-time.

navigator at the top of the page to create a edit an existing one from the list.

The same checks the editor runs

Canvas decides what can bind

The candidate list is the editor's own.

An agent cannot bind anything a site builder could not.

Two rules it will not bend

An `enum` prop takes a static value, never a field.

A `required` prop will not bind to an optional one.

04

The agent **builds a page.**

"Build the welcome page from its tree in the handoff, with the copy the handoff gives. Then publish."

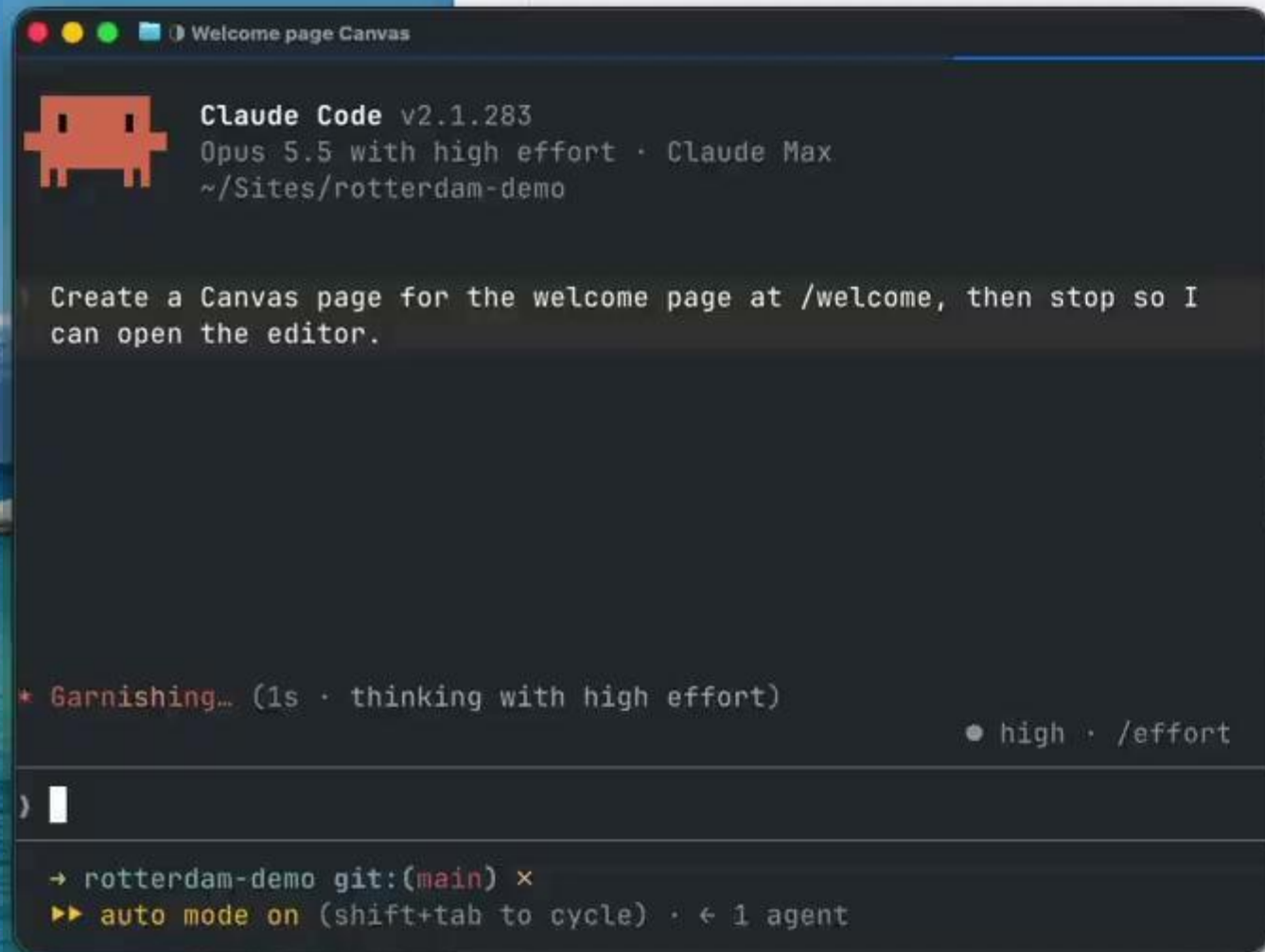
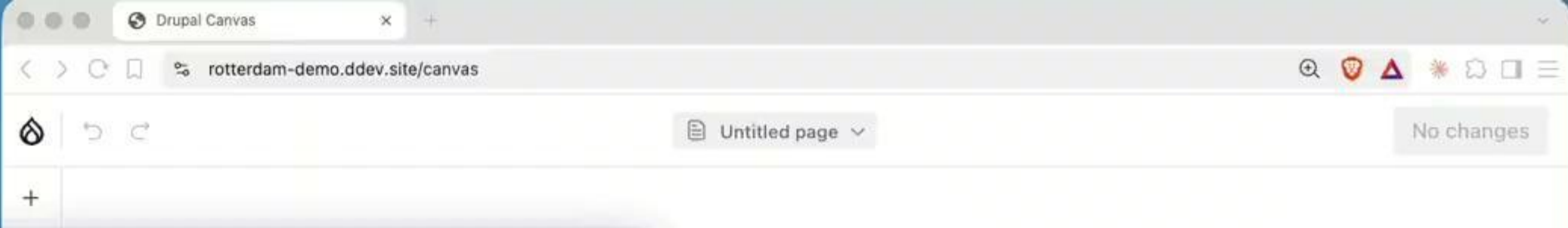
Same calls, a different target

```
create_page          {page, "Welcome", published} → page 1
add_component        {page 1, sdc...site-header} → uuid # ×5
update_component_props {uuid, {...the copy from the handoff}} # ×5
publish_auto_saves    {page 1} → live
# no bind_component_props — a page has no host entity
```

Every value is a literal from the handoff.

There is nothing to bind to.

All canvas_* · target_type defaults to page · published is intent until the last call runs



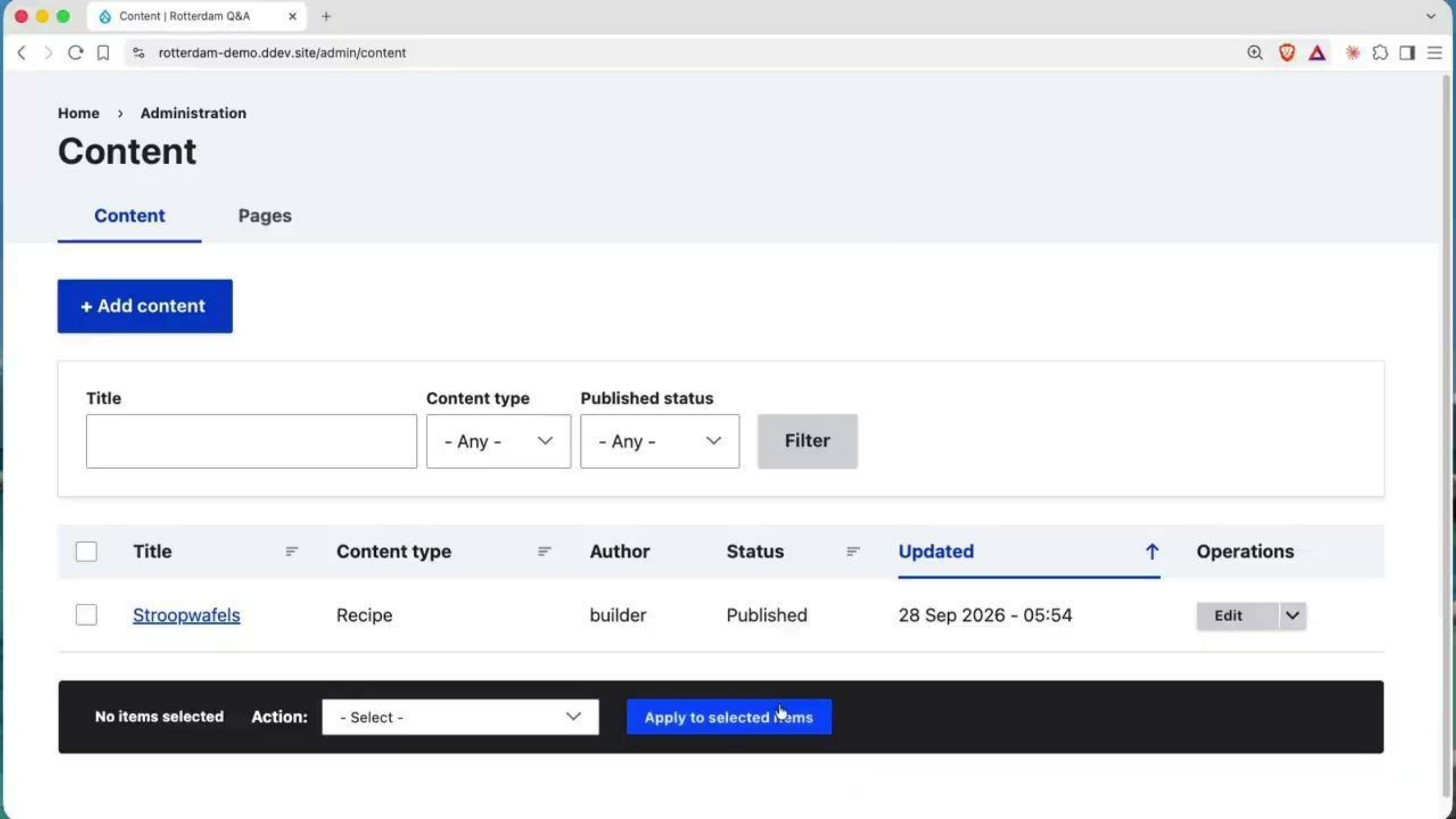
elcome to Drupal Canvas! This tool allows you to
and manage your content experiences with
you can add components, customize layouts,
view your changes in real-time.

navigator at the top of the page to create a
edit an existing one from the list.

THE REVEAL

Open the page.

No agent. Just the site, and what it built.



Content

Content

Pages

+ Add content

Title

Content type

Published status

Filter



Title



Content type



Author

Status



Updated



Operations



[Stroopwafels](#)

Recipe

builder

Published

28 Sep 2026 - 05:54

Edit



No items selected

Action:

- Select -



Apply to selected items

It renders.

And it is still **human-editable**.

Not throwaway HTML that someone now has to manage.

A real Canvas page, and a real content template behind the recipe.

Open either one in the editor and keep going.

TO CLOSE

MCP, gaps, takeaways

Where the other transport comes back, what still hurts, and what to keep.

Everything you just saw is already on MCP

```
canvas_create_page    →  tool_api__canvas_create_page  
tool_belt:field_add   →  tool_api__tool_belt__field_add
```

Same plugins, no extra code.

The MCP tools list hands them back under the names above.

One `mcp_tool_config` entity per tool, already shipped in the module.

Honest gaps

Canvas has no public API

Everything the bridge touches is internal, so it is coupled to a Canvas version on purpose.

MCP auth is not free

On the CLI you pick the user that STDIO runs as.
Over HTTP you need OAuth, and [mcp_server_oauth](#) is still alpha.

Four things to take home

1

Auto-save is the way in

Ride the editor's own auto-save and publish path, and you inherit its rules for free.

2

Build at the tool layer

Build it once at the Tool API layer and the CLI, MCP and the AI module all get it. You do not pick a transport.

3

Skills travel with the tools

Require the module and the agent gets the workflow, not just the tools.

4

Verification is first-class

The agent checks its own output instead of guessing, and the same command gates CI.

And a person can open everything it built and carry on

Thank you

THE MODULES

canvas_tools

tool · tool_belt

mcp_server

SURVEY



FIND ME

mglaman.dev

drupal.org/u/mglaman