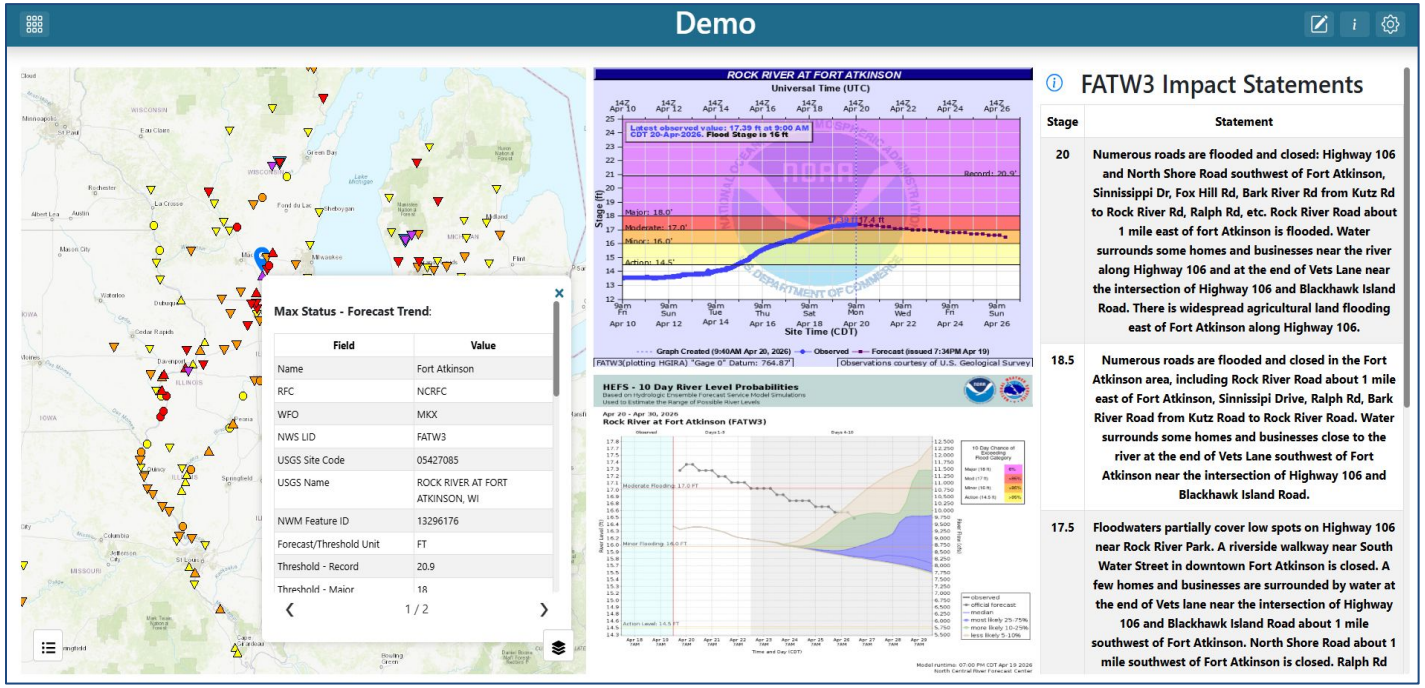
TethysDash

A No-Code Dashboard Builder for the Tethys Platform

Introduction & Feature Overview

What is TethysDash?

- A no-code / low-code dashboard builder
- Runs as a Tethys Platform app (Django + React)
- Users compose dashboards from draggable widgets
- Widgets are powered by out of the box visualizations and a python based plugin system for customized visualizations
- Supports charts, maps, tables, images, text, and custom components
- Built-in user interactivity through Variable Inputs
- Permissions at dashboard, visualization, and group level



Who Is It For?

Scientists & Analysts	Water Resource Agencies	Research Teams	Developers
Visualize model output, forecasts, and observational data without writing frontend code.	Publish operational dashboards for reservoirs, streamflow, water quality, and alerts.	Share interactive results with collaborators and stakeholders.	Build reusable plugins once, let non-technical users assemble dashboards.

The same app serves both builders and viewers — no separate tooling.

Core Concepts

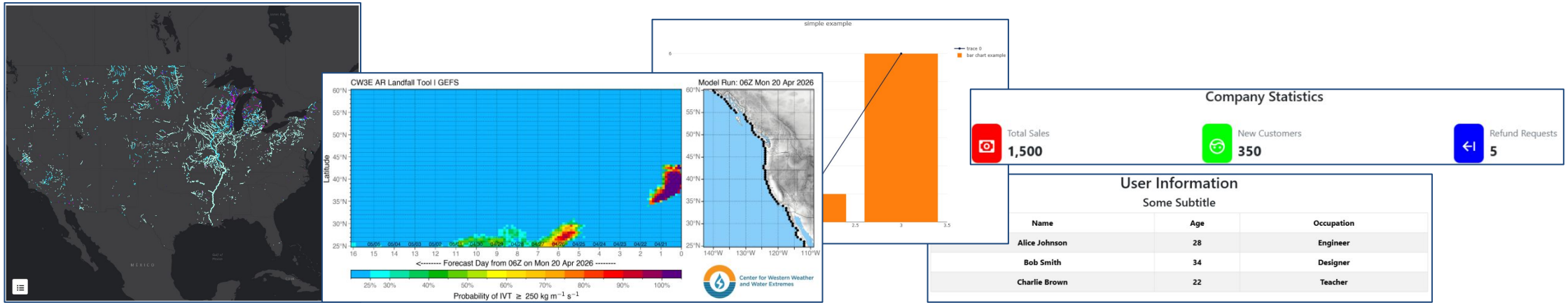
Dashboard	Grid Item	Plugin (Visualization)
A named workspace containing a grid of visualizations. Has owner, permissions, and sharing settings.	A single widget on a dashboard. References a plugin and its arguments. Draggable and resizable.	An external Python package that produces data for one widget. Registered through Intake.

Dashboard → contains many Grid Items → each Grid Item uses one Plugin.

Visualization Types

One plugin system, many renderers

`plotly`	Interactive charts — line, bar, scatter, heatmap, 3D, etc.
`table`	Tabular data with titles, subtitles, and sortable columns.
`text`	Formatted rich text blocks for labels and narration.
`variable_input`	User-facing controls that drive other widgets.
`map`	Geospatial layers via OpenLayers — base maps, vector, raster.
`card`	KPI-style metric cards with color, label, value, and icon.
`image`	Static or dynamically generated image URLs.
`custom`	Fully custom React components via Module Federation.



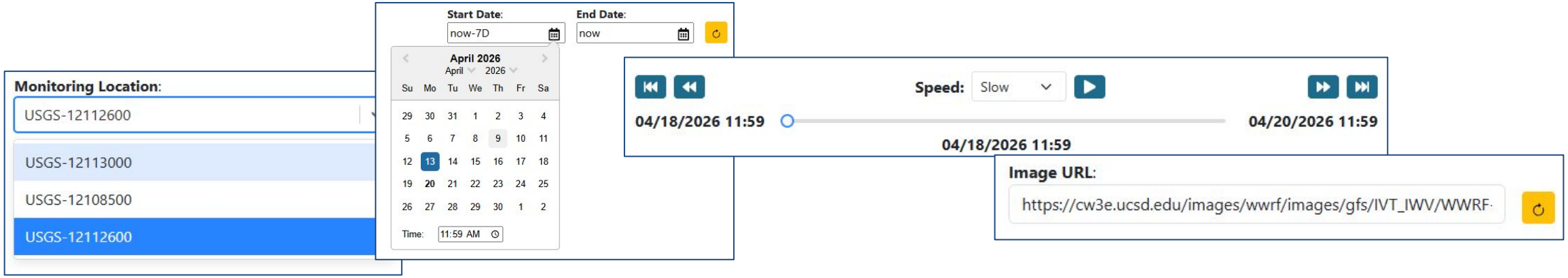
Variable Inputs

- A special widget type that captures user input
- Each variable input has a name (e.g., "Selected Site")
- Other widgets reference that name in their arguments
- When the user changes the input, dependent widgets auto re-fetch
- Makes a single dashboard dynamic without any code
- Maps can also update variable inputs through clicking events

Variable Input Types

Choose the right control for the data

`text`	Free-form text entry	`number`	Numeric entry with validation	`checkbox`	Boolean on / off
`date`	Single date picker	`date-range`	Start + end date picker	`dropdown`	Select one from a list
`slider`	Numeric range selector	`csv-uploader`	User-uploaded CSV feeds other widgets	`auto-generated`	Built directly from existing plugin args



Same data flow for every input type — just pick the UX that fits.

Permissions

Three layers, composed together

1. Dashboard permissions

Per dashboard, per user or group. Roles: admin, editor, viewer. Controls who can see and who can edit a specific dashboard.

2. Visualization permissions

Per plugin type, per user or group. Controls which plugins a user is even allowed to add to any dashboard.

3. Permission groups

Owner-managed user groups with their own admin / editor / member roles. Assign dashboards or visualizations to a group in one step.

Layers stack: a user needs the right dashboard role AND the right visualization permission.

TethysDash Core Tutorial

Building a dashboard from scratch

https://tethysdash.readthedocs.io/en/latest/tutorials/geoglows_demo.html

The Plugin System

How new visualizations get added

- Plugins live in external Python packages
- Each plugin subclasses TethysDashPlugin
- Must implement run() → returns data for its visualization type
- May implement send_update() → stream progress via WebSocket
- The plugin's `type` determines the frontend renderer used
- The plugin's `args` determine the arguments rendered in the TethysDash for the application
- Install a new plugin package via pip→ it appears in TethysDash

Minimal plugin shape

```python
from tethysapp.tethysdash.plugin_helpers import TethysDashPlugin
import plotly.express as px
import json

class PlotExample(TethysDashPlugin):
    name = 'plot_example'
    args = {"continent": "text"}
    group = "Example"
    label = "Example Plot"
    type = "plotly"
    tags = [
        "example",
        "plotly",
    ]
    description = "An example plugin for the plotly visualization"

    def run(self):
        df = px.data.gapminder().query(f"continent == '{self.continent}'")
        fig = px.line(df, x="year", y="lifeExp", color="country", symbol="country")
          return json.loads(fig.to_json())
```

TethysDash Plugin Tutorial

Building a plugin from scratch

https://tethysdash.readthedocs.io/en/latest/tutorials/geoglows_demo_part2.html

Questions?

Let's build something together.