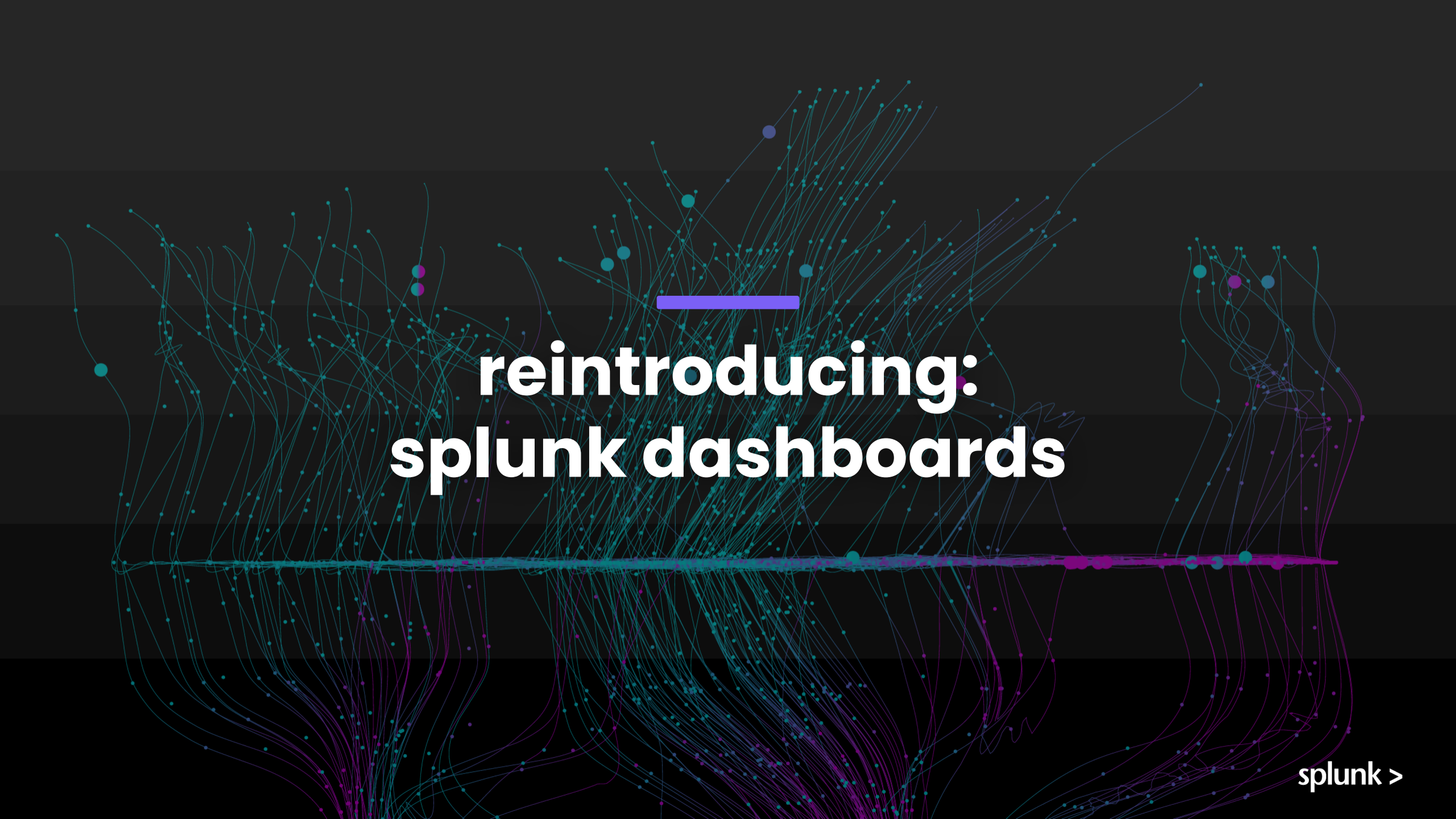

reintroducing:
splunk dashboards
splunk >

Forward Looking Statements

During the course of this presentation, we may make forward-looking statements regarding future events or plans of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to use and that actual events or results may differ materially. The forward-looking statements made in this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, it may not contain current or accurate information. We do not assume any obligation to update any forward-looking statements made herein.

In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionalities described or to include any such feature or functionality in a future release.

splunk >

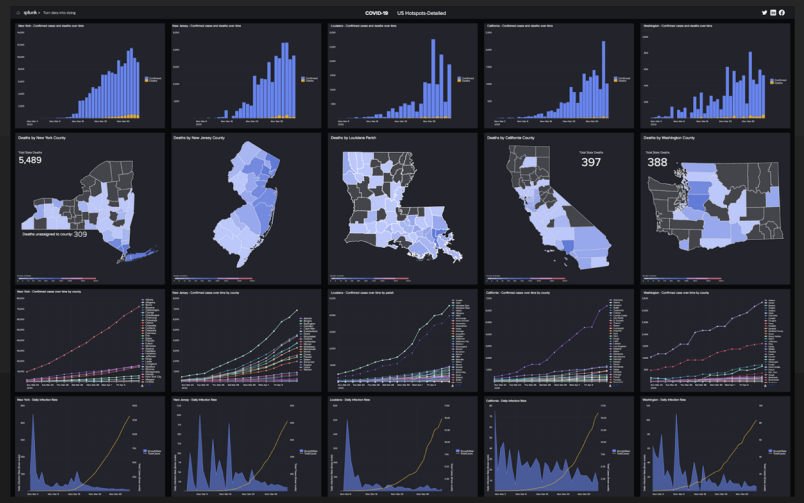
speakers

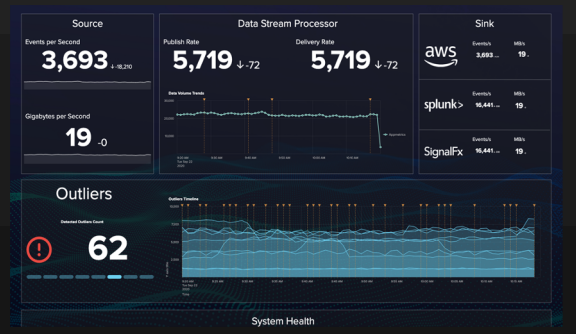
Miranda Luna
Product Manager

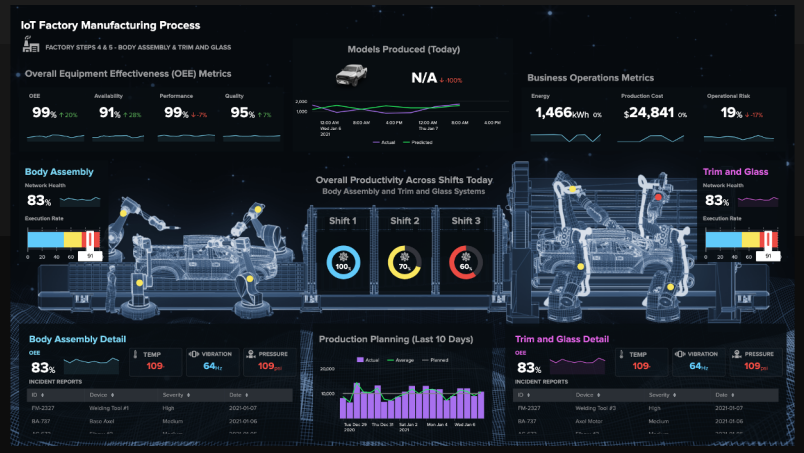
Lizzy Li
Product Manager

agenda

- introducing splunk dashboard studio
  - features overview
  - roadmap
- extending the underlying dashboard framework
  - custom layouts, visualizations, datasources, inputs, event handlers
  - publish to public URL
- demo!
- helpful resources

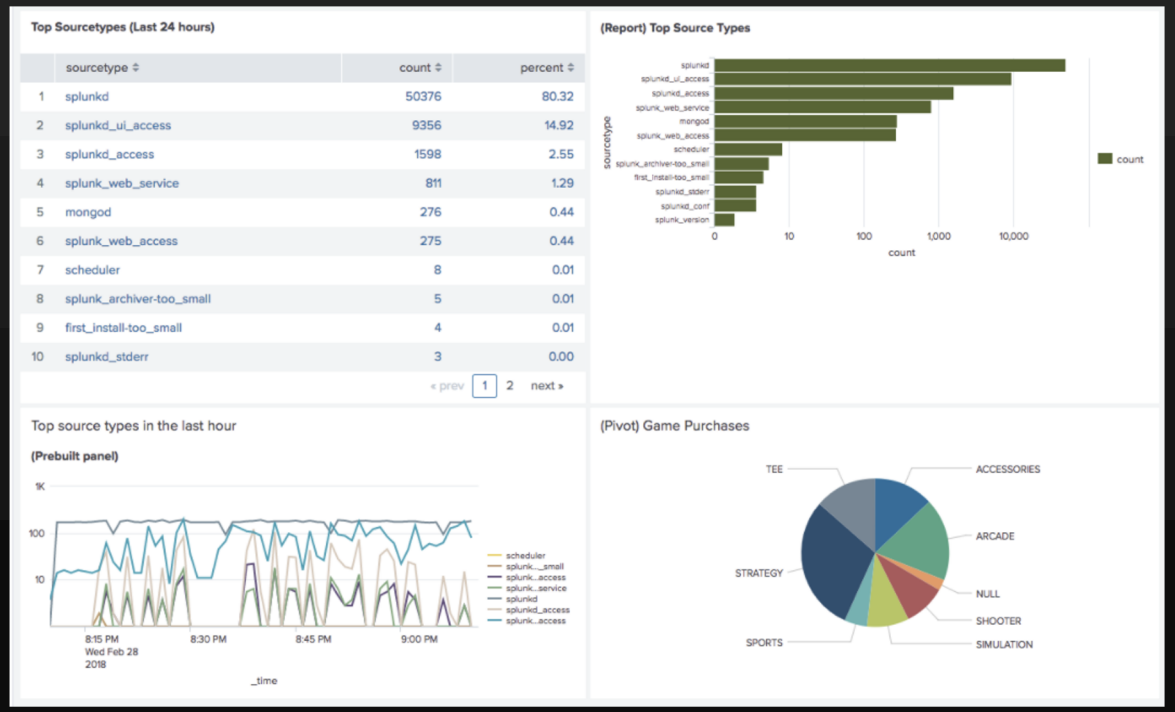




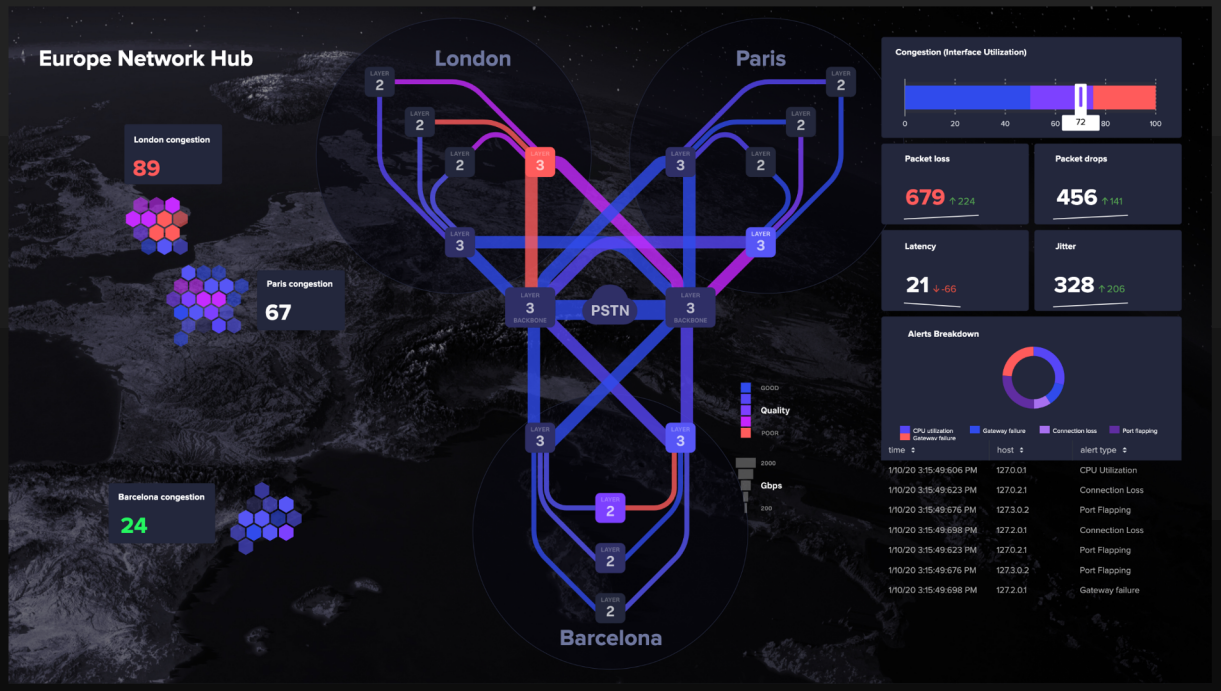


introducing splunk dashboard studio

simple xml



dashboard studio



splunk dashboard studio

- previously known as the splunk dashboard beta app on splunkbase
- now available in search & reporting, starting with splunk cloud 8.1.2103 and splunk enterprise 8.2

key dashboard studio ga features

- flexible layout options
  - absolute layout for free placement, sizing, and layering of objects (pixel-perfect)
  - grid layout for neat and quick alignment of charts, still with resizing capabilities
- native support for new visualizations
  - images, icons, text boxes, shapes, lines, and choropleth svgs
- on-demand png/pdf export that maintains the dashboard look & feel
- improved ux for doing more configurations in the ui
  - you can view the source code for a specific data source, input, or viz directly in the ui configuration panel

key dashboard studio roadmap features

- advanced token and drill down functionality
- more enhanced visualizations
- more ui for viz, data source, and inputs configurations
- scheduled dashboard export

developing with the framework

@splunk/dashboard-core

semantically versioned packages regularly shipped to npm at release

https://www.npmjs.com/package/@splunk/dashboard-core

leverages modern front-end technologies (react/ redux)

modular

all dependencies resolved at build time

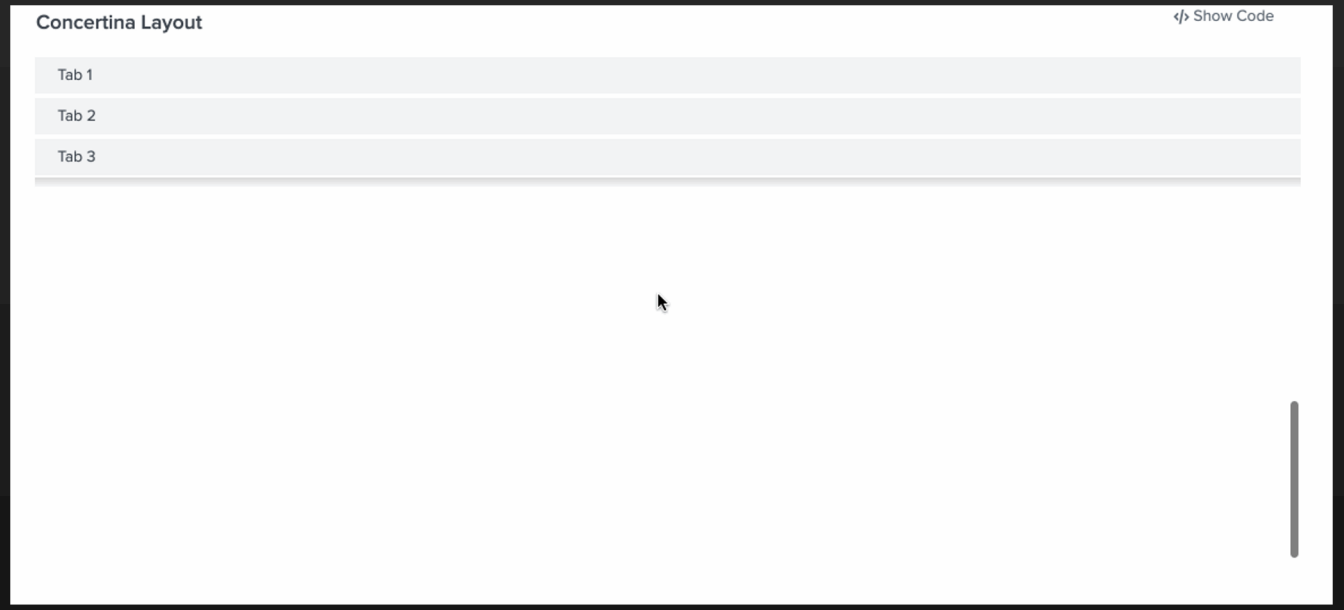
so easy to extend & plug in custom layouts, visualizations, etc.

platform agnostic

build for splunk enterprise, splunk cloud, or use as a single pane of glass into multiple datastores

custom layouts

```jsx
/* eslint-disable react/no-array-index-key */
import React, { useMemo } from 'react';
import BaseLayout from '@splunk/dashboard-layouts/BaseLayout';
import Concertina from '@splunk/react-ui/Concertina';

const panelStyle = { padding: '20px' };

/**
 * Custom layout components
 * @param {*} props
 */
const CustomLayout = ({
    containerWidth,
    containerHeight,
    layoutStructure,
    renderLayoutItem,
}) => {
    const concertinaStyle = useMemo(
        () => ({
            width: containerWidth,
            height: containerHeight,
        }),
        [containerWidth, containerHeight]
    );

    return (
        <Concertina style={concertinaStyle}>
            {layoutStructure.map(({ item, title }, idx) => (
                <Concertina.Panel
                    title={title}
                    style={panelStyle}
                    key={`${title}_${idx}`}
                >
                    {renderLayoutItem(item, { height: 400 })}
                </Concertina.Panel>
            ))}
        </Concertina>
    );
};

CustomLayout.propTypes = {
    ...BaseLayout.propTypes,
};
CustomLayout.defaultProps = {
    ...BaseLayout.defaultProps,
};

export default CustomLayout;
```



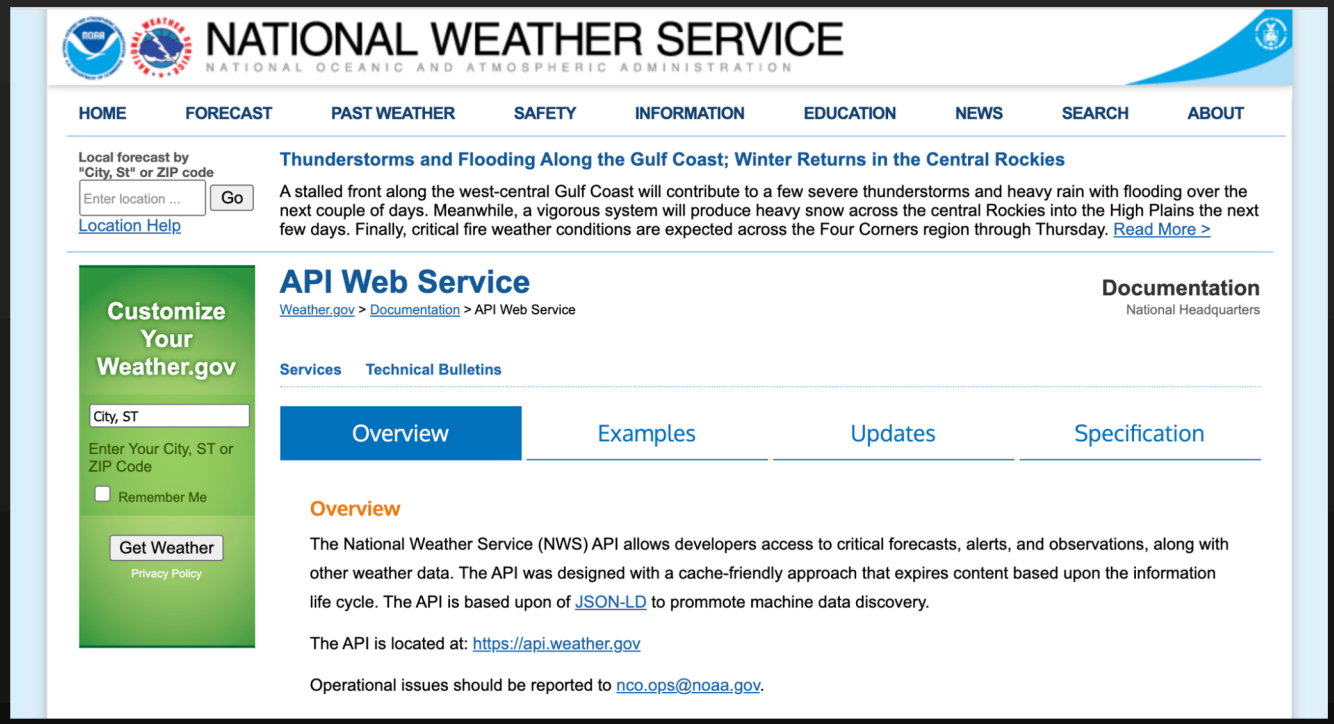
custom visualizations

```jsx
import React, { useMemo } from 'react';
import BaseVisualization from '@splunk/dashboard-visualizations/common/BaseVisualization';
import JSONTree from '@splunk/react-ui/JSONTree';

/**
 * A React component that renders its props
 * @param {*} props
 */
const CustomViz = (props) => {
    const { width, height } = props;
    const style = useMemo(
        () => ({
            height,
            width,
            overflow: 'auto',
        }),
        [width, height]
    );
    return (
        <div style={style}>
            <JSONTree json={props} />
        </div>
    );
};

CustomViz.propTypes = {
    ...BaseVisualization.propTypes,
};
CustomViz.defaultProps = {
    ...BaseVisualization.defaultProps,
};

/**
 * data contract for this visualization
 */
CustomViz.dataContract = {
    // required data sources
    requiredDataSources: ['primary'],
    // initial requestParams for each data sources
    initialRequestParams: {
        primary: {
            offset: 0,
            count: 50,
        },
    },
};
```

```jsx
/**
 * visualization descriptor
 */
CustomViz.config = {
    dataContract: {},
    editorConfig: [],
    optionsSchema: {},
    key: 'viz.custom',
    name: 'Custom Viz',
    icon: <div />,
    category: 'Comparisons',
};

/**
 * visualization can be added through the toolbar
 */
CustomViz.includeInToolbar = true;

/**
 * can switch to this visualization through the editor viz switcher
 */
CustomViz.includeInVizSwitcher = true;

export default CustomViz;
```

custom datasources

```
import DataSource from '@splunk/datasources/DataSource';
import DataSet from '@splunk/datasource-utils/DataSet';
import fetch from 'node-fetch';

export default class WeatherDatasource extends DataSource {
    // Execute the search during setup (fetch the weather, save a promise)
    setup() {
        this.api = fetch(
            'https://api.weather.gov/gridpoints/TOP/31,80/forecast'
        );
    }

    // Execute the status/results observable flow
    request() {
        return (observable) => {
            this.api
                // Process the response into JSON
                .then((res) => {
                    return res.json();
                })
                .then((res) => {
                    // Create a DataSet
                    const data = DataSet.fromJSONArray(
                        [
                            { name: 'name' },
                            { name: 'startTime' },
                            { name: 'endTime' },
                            { name: 'temperature' },
                            { name: 'windSpeed' },
                            { name: 'windDirection' },
                            { name: 'shortForecast' },
                        ],
                        res.properties.periods
                    );
                    // Populate metaData
                    const meta = {
                        status: 'done',
                        totalCount: res.properties.periods.length,
                    };

                    // Push data to the observer
                    observable.next({ data, meta });
                    // Inform the observer will be no more data
                    observable.complete();
                });

            // Do nothing in observable teardown
            return () => {};
        };
    }
}
```



DataSource is a javascript module that provides the data that powers a dashboard

You can create a custom datasource to access data from virtually any resource

Only requirement is the data sent to the search observer is returned in the form of a DataSet and a properly formatted meta object

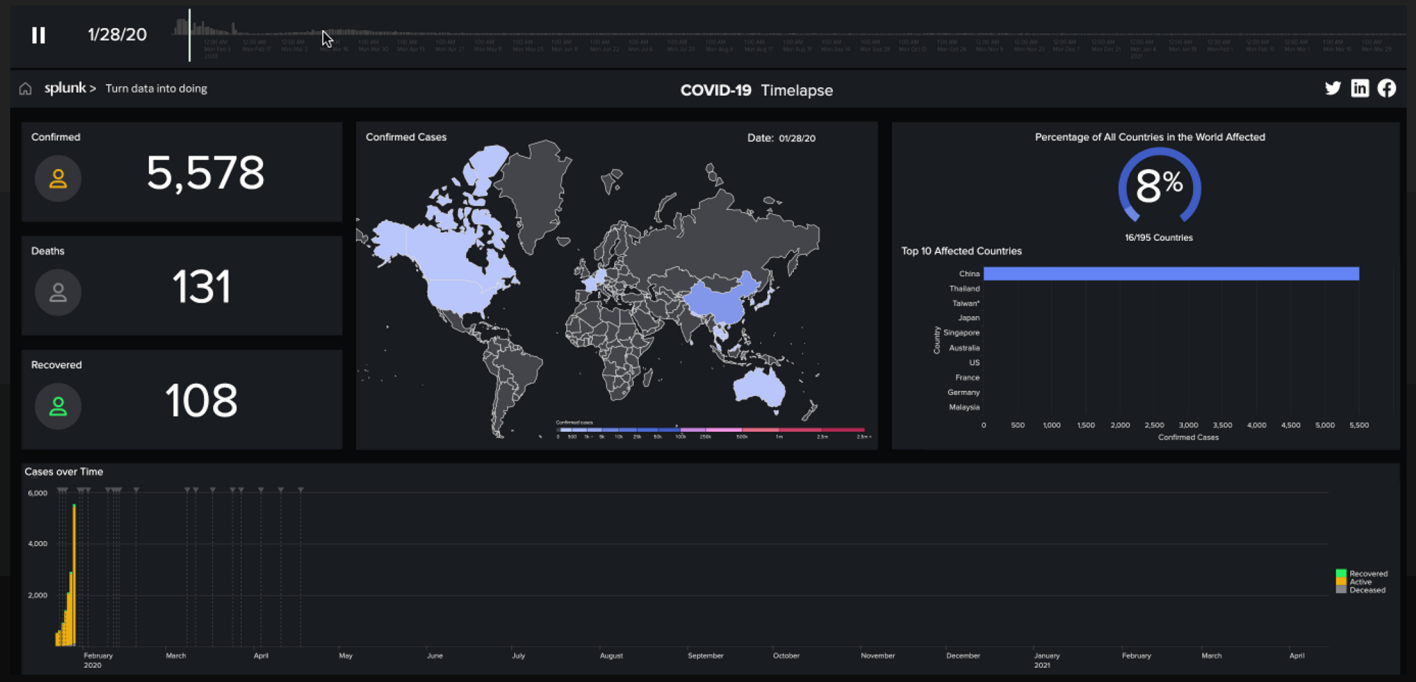
event handlers

```
class TableClickHandler {
    constructor(options) {
        this.options = options;
    }

    canHandle(event) {
        // only handle table cell.click event on particular field
        return (
            event.type === 'cell.click' &&
            event.payload.fieldValue === this.options.field
        );
    }

    handle(event) {
        return [
            {
                type: 'setToken',
                payload: {
                    tokens: {
                        title: `Click on field ${this.options.field} cell value: ${event.payload.cellValue}`,
                    },
                },
            },
        ];
    }
}

export default TableClickHandler;
```

an eventhandler is a javascript module that handles dispatched action events with the following two functions

- **canHandle(event)** - Returns a boolean value. 'True' indicates that the current handler is capable of handling the current action event, while 'false' indicates that it is not.
- **handle(event)** - Handles an action event and returns a list of actions. The returned actions will be executed by the dashboard one after the other.

other custom components



how it all comes together

full credit to atef kouki!

Experimental

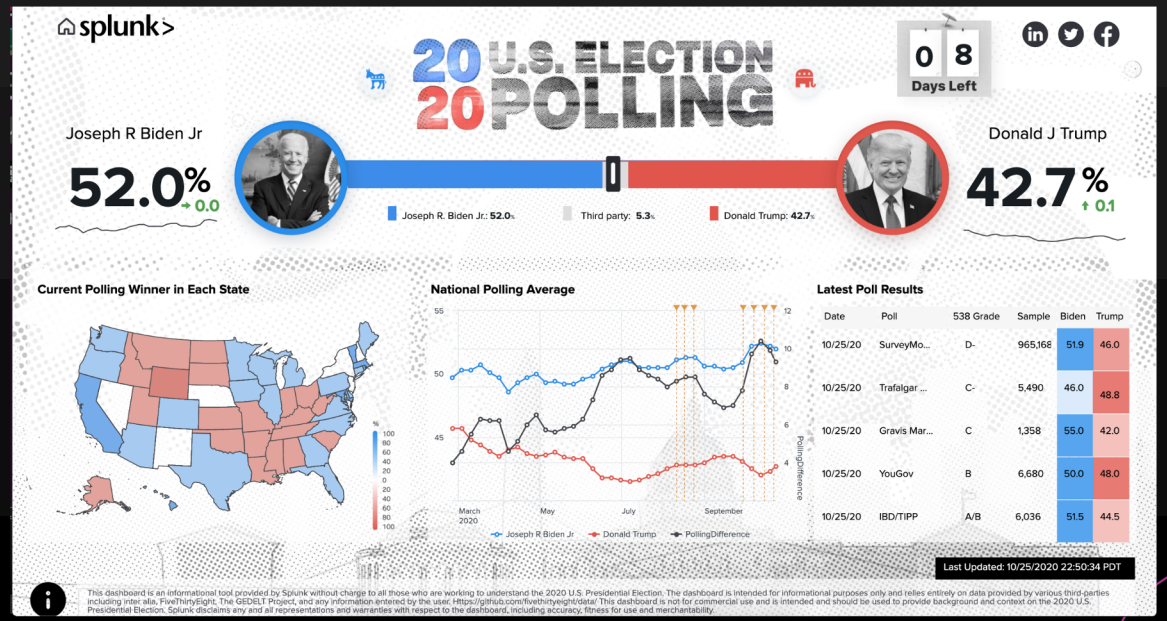
publish to public url

Not officially supported

publish splunk dashboard to public url

serverless api endpoints
access to search results (public, unauthenticated, cached)
deployed to cdn
publisher cli generates boilerplate code, dashboard code, optimized assets, data snapshots

https://covid-19.splunkforgood.com/



https://election2020.splunkforgood.com/


demo

resources

dashboard studio editing experience demo
https://www.youtube.com/watch?v=hI8MBXR6UQ0

svg choropleth deep dives
https://www.splunk.com/en_us/blog/platform/painting-with-data-choropleth-svg.html
https://www.splunk.com/en_us/blog/platform/advanced-painting-with-data-choropleth-svg.html

npm package for dashboard rendering engine
https://www.npmjs.com/package/@splunk/dashboard-core

publish dashboard to public url
https://ideas.splunk.com/ideas/EID-I-865
https://github.com/splunk/dashpub
conf 2020 session recording | slides

custom timeslider component
conf 2019 session recording | slides

medical image viewer application
https://www.splunk.com/en_us/blog/platform/splunk-at-the-service-of-medical-staff.html
leverages https://www.npmjs.com/package/react-image-gallery
source code https://drive.google.com/drive/folders/1pR5SA1L2QhmXwys2sq8LVIcxC0U17u9W

ideas.splunk.com

dashboard-studio@splunk.com