



# How do I add a Component to the sidebar?

You can add a component to `st.sidebar` using the `with` syntax. For example:

```
with st.sidebar:  
    my_component(greeting="hello")
```

In fact, you can add your component to *any* layout container (eg `st.columns`, `st.expander`), using the `with` syntax!

```
col1, col2 = st.columns(2)  
with col2:  
    my_component(greeting="hello")
```

## Was this page helpful?

Yes    No

[Suggest edits](#)



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

**Next:** My Component seems to be stuttering...how do I fix that? [→](#)

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



## Advanced features

This section gives you background on how different parts of Streamlit work.

### Theming

This section provides

- [primaryColor](#)
- [backgroundcolor](#)
- [secondarybackgroundcolor](#)
- [textcolor](#)
- [font](#)
- [base](#)

### Streamlit configuration

When you install Streamlit, a command-line (CLI) tool gets installed as well. The purpose of this tool is to run

Streamlit apps, change Streamlit configuration options, and help you diagnose and fix issues.

- [What is the command-line interface \(CLI\)?](#)

- [How to run Streamlit apps from the CLI?](#)

- [View documentation from the CLI](#)

- [Clear cache](#)

from the  
CLI

- How to  
set  
configuration  
options?
  - View all  
configuration  
options
- 

Optimize  
performance  
with  
st.cache

- Streamlit provides a caching mechanism that allows your app to stay performant even when loading data from the web, manipulating large datasets, or performing expensive computations. This is done with the @st.cache decorator.
- What is  
st.cache?
  - Basic  
usage
  - Caching  
when the  
function  
arguments  
change?
  - Caching  
when the  
function  
body  
changes
  - Caching  
when an  
inner  
function  
changes
  - Use  
caching  
to speed  
up your  
app  
across  
users
  - How to  
mutate

mutate  
cached  
values?

- Advanced caching
- 

Streamlit

provides two experimental primitives to memoize function executions and store singleton objects.

@st.experimental\_memo

is used to store expensive computation which can be "cached" or "memoized" in the traditional sense.

@st.experimental\_singleton

is a key-value store that's shared across all sessions of a Streamlit app.

It's great for storing heavyweight singleton objects across sessions (like

TensorFlow/Torch/Keras sessions and/or database connections).

- Problems with st.cache
  - @st.experimental\_memo
  - Properties of @st.experimental\_memo
  - @st.experimental\_singleton
  - How @st.experimental\_singleton compares to @st.cache
  - Which to use: memo or singleton?
- 

Session State is a way to share variables between reruns, for each user session. In addition to the ability to store and persist state, Streamlit also exposes the ability to manipulate state using Callbacks.

- What is Session State?
- How to initialize Session State items?
- How to read and update

- How to use callbacks in Session State?
- How to use args and kwargs in callbacks?
- How to use callbacks in forms?
- How is Session State related to Widget State?
- Caveats and limitations

---

## Pre-release features

At Streamlit, we like to move quick while

keeping things stable. In our latest effort to move even faster without sacrificing stability, we're offering our bold

- Nightly releases
- Beta and experimental features

and fearless  
users two ways  
to try out  
Streamlit's  
bleeding-edge  
features.

---

## Advanced notes on widget behavior

Widgets are magical and often work how you want. But they can have surprising behavior in some situations. This section provides a high-level, abstract description of widget behavior, including some common edge-cases.

## Working with timezones

Working with timezones can be tricky. This section provides a high-level description of how to handle timezones in Streamlit to avoid unexpected behavior.

- [Overview](#)
- [How Streamlit handles timezones](#)
- [datetime instance without a timezone \(naive\)](#)
- [datetime instance with a timezone](#)

Was this page helpful?

 Yes     No

 [Suggest edits](#)

---

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[!\[\]\(96cc62f861fdd6e50510c0224a756dff\_img.jpg\) Previous:](#) API reference

[Next:](#) Theming [!\[\]\(fa6f3af6bfa46c5d4a2d362681095beb\_img.jpg\)](#)

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.

## Documentation

*search*

Search

- *description*

### Streamlit library

- Get started  
*add*
- API reference  
*add*
- Advanced features  
*add*
- Components  
*add*
- Changelog
- Cheat sheet

- *cloud*

### Streamlit Cloud

- Get started  
*add*
- Trust and Security
- Release notes*open in new*
- Troubleshooting

- *school*

### Knowledge base

- Tutorials  
*add*
- Using Streamlit
- Streamlit Components
- Installing dependencies
- Deployment issues

- *Home*/
- *Knowledge base*/
- *Using Streamlit*/
- *How to animate elements?*

## **How to animate elements?**

8

Let's combine some of the things you've learned to create compelling animations in your app.

```
progress_bar = st.progress(0)
status_text = st.empty()
chart = st.line_chart(np.random.randn(10, 2))

for i in range(100):
    # Update progress bar.
    progress_bar.progress(i + 1)

    new_rows = np.random.randn(10, 2)

    # Update status text.
    status_text.text(
```

```
'The latest random number is: %s' % new_rows[-1, 1])  
  
# Append data to the chart.  
chart.add_rows(new_rows)  
  
# Pretend we're doing some computation that takes time.  
time.sleep(0.1)  
  
status_text.text('Done!')  
st.balloons()
```

Was this page helpful?

[\*thumb\\_up\* Yes](#) [\*thumb\\_down\* No](#)  
[edit](#) [Suggest edits](#)  
[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: Using Streamlit](#) [Next: Append data to a table or chart](#) →

---

[Home](#) [Contact Us](#) [Community](#)



Copyright © 2022, Streamlit Inc.



# API reference

Streamlit makes it easy for you to visualize, mutate, and share data. The API reference is organized by activity type, like displaying data or optimizing performance. Each section includes methods associated with the activity type, including examples.

Browse our API below and click to learn more about any of our available commands! 🎈

## Display almost anything

### st.write

Write arguments to the app.

```
st.write("Hello **world**!")  
st.write(my_data_frame)  
st.write(my_mpl_figure)
```

### Magic

Any time Streamlit sees either a variable or literal value on its own line, it automatically writes that to your app using

`st.write`

```
"Hello **world**!"  
my_data_frame  
my_mpl_figure
```

## Text elements

### Tincidunt lobortis

Feugiat vivamus at augue eget arcu dictum. Sed risus pretium quam vulputate. Cursus in hac habitasse

platea dictumst. Aliquam ultrices sagittis orci a.

## Non diam phasellus vestibulum

Vel quam elementum pulvinar etiam. Blandit volutpat maecenas volutpat blandit aliquam. Est sit amet

### Markdown

Display string formatted as Markdown.

```
st.markdown("Hello **world**!")
```

**This is a title**  
**This is a title**  
**This is a title**

### Title

Display text in title formatting.

```
st.title("The app title")
```

**This is a header**  
**This is a header**  
**This is a header**

### Header

Display text in header formatting.

```
st.header("This is a header")
```

**This is a subheader**  
**This is a subheader**

# This is a subheader

## Subheader

Display text in subheader formatting.

```
st.subheader("This is a subheader")
```

## This is a caption

## This is a caption

This is a caption.

## Caption

Display text in small font.

```
st.caption("This is written small caption text")
```

```
if current_frame.f_back is not None:  
    lines = inspect.getframeinfo(current_frame.f_back)[3]  
else:  
    lines = None  
  
if not lines:  
    warning("`show` not enabled in the shell")  
    return
```

## Code block

Display a code block with optional syntax highlighting.

```
st.code("a = 1234")
```

## Preformatted text

Write fixed-width and preformatted text.

```
st.text("Hello world")
```

$$\mathbf{E}' = \nabla \times \mathbf{B} - 4\pi j$$

$$G_{\mu\nu} + \Lambda g_{\mu\nu} = \frac{8\pi G}{c^4} T_{\mu\nu}$$

## LaTeX

Display mathematical expressions formatted as LaTeX.

```
st.latex("\int a x^2 \,dx")
```

## Data display elements

Country	1.1.2009	1.1.2013	1.1.2017	1.1.2016	1.1.2014	1.1.2015
Brazil	16.5646	17.3449	17.4430	17.6581	20.3672	19.2388
China	58.3407	60.6909	63.9427	68.4626	74.7479	80.4459
France	23.4085	26.6540	25.2027	25.3211	26.7809	25.4474
Germany	22.0777	24.2415	25.3408	24.8436	23.9885	25.2363
India	50.1427	50.0158	51.3677	52.2677	49.4983	49.3096
Indonesia	7.3837	8.0549	7.6509	8.0770	7.9255	8.2192
Mexico	6.6326	7.0989	7.3942	8.0449	8.6041	9.1411

## Dataframes

Display a dataframe as an interactive table.

```
st.dataframe(my_data_frame)
```

	A	B	C	D	E
0	1.000000	1.329212	nan	-0.316280	-0.990810
1	2.000000	-1.070816	-1.438713	0.564417	0.295722

2	3.000000	-1.626404	0.219565	0.678805	1.889273
3	4.000000	0.961538	0.104011	-0.481165	0.850229
4	5.000000	1.453425	1.057737	0.165562	0.515018

## Static tables

Display a static table.

```
st.table(my_data_frame)
```

Temperature	Wind	Humidity
70 °F	9 mph	86%
↑ 1.2 °F	↓ -8%	↑ 4%

## Metrics

Display a metric in big bold font, with an optional indicator of how the metric changed.

```
st.metric("My metric", 42, 2)
```

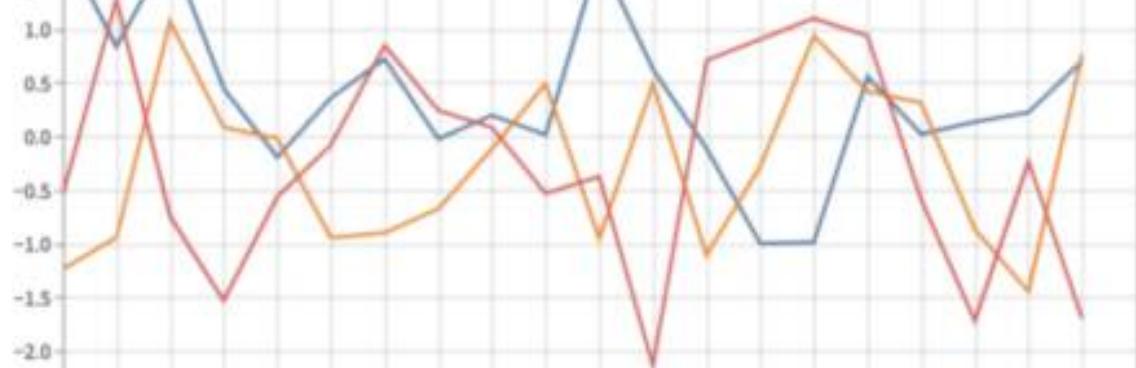
```
  "init-param" : {...}
}
* 1 : {
    "servlet-name" : "cofaxCDS"
    "servlet-class" : "org.cofax.cds.CDSServlet"
    * "init-param" : {
        "configGlossary:installationAt" : "Philadelphia, PA"
        "configGlossary:adminEmail" : "ksm@pobox.com"
```

## Dicts and JSON

Display object or string as a pretty-printed JSON string.

```
st.json(my_data_frame)
```

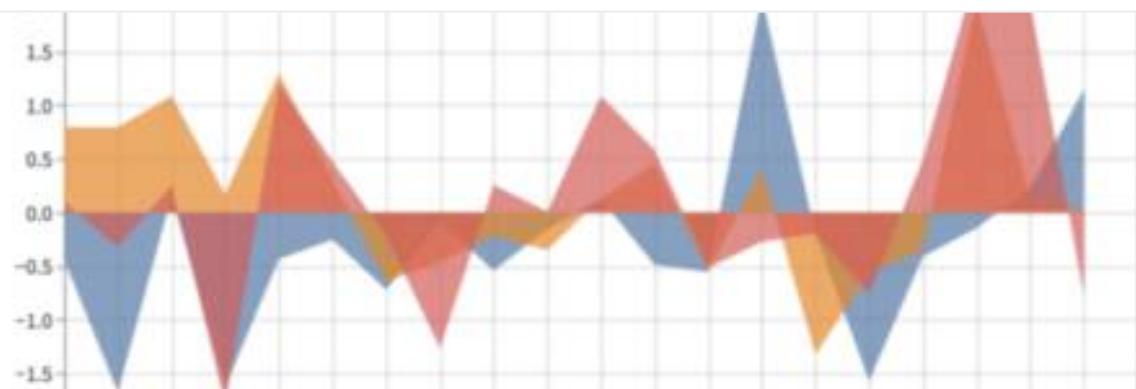
## Chart elements



## Simple line charts

Display a line chart.

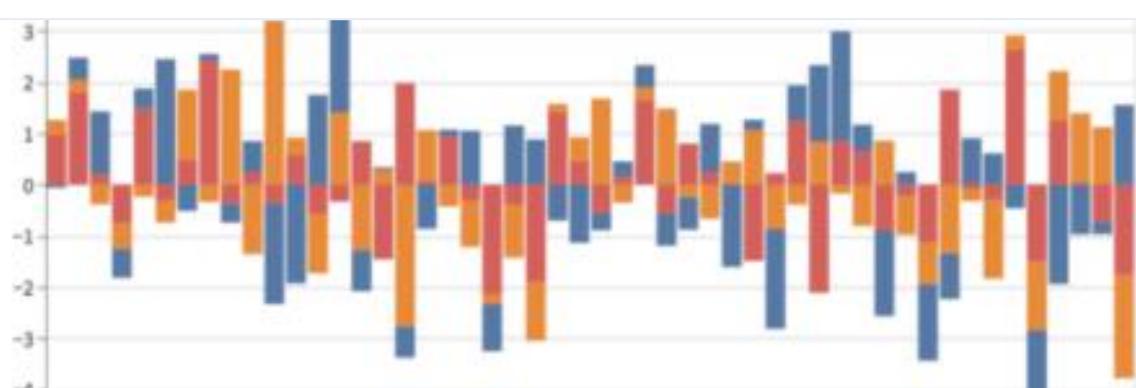
```
st.line_chart(my_data_frame)
```



## Simple area charts

Display an area chart.

```
st.area_chart(my_data_frame)
```



## Simple bar charts

Display a bar chart.

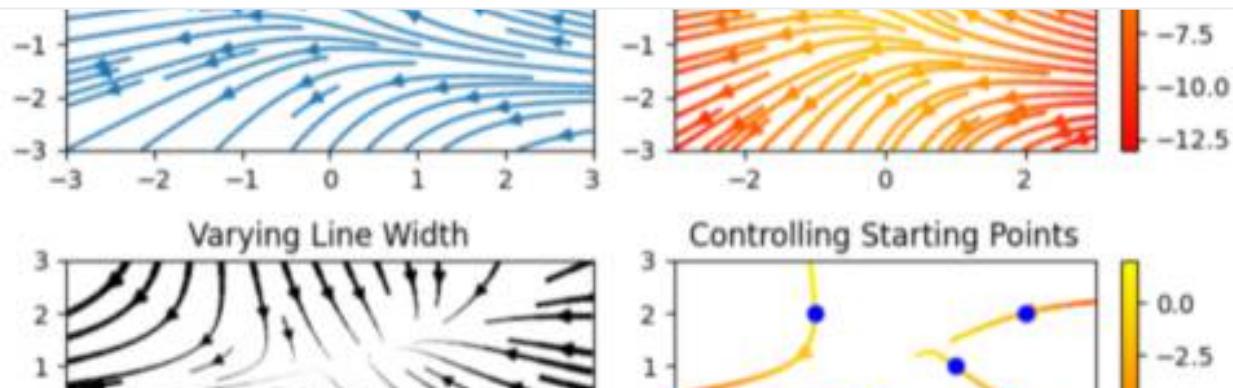
```
st.bar_chart(my_data_frame)
```



## Scatterplots on maps

Display a map with points on it.

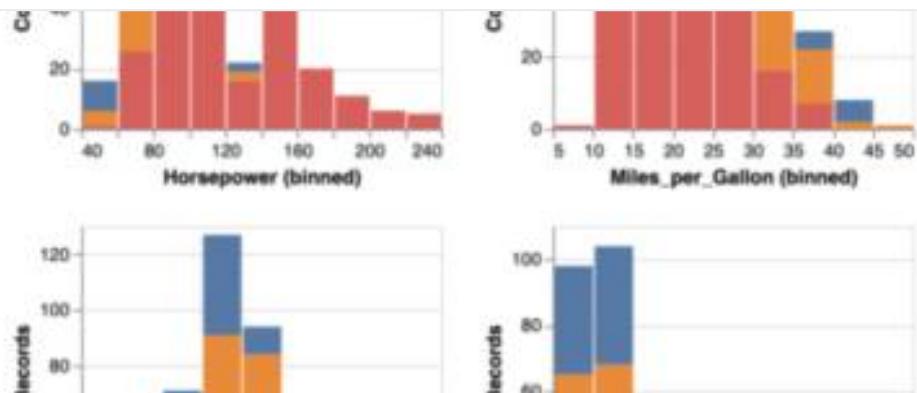
```
st.map(my_data_frame)
```



## Matplotlib

Display a matplotlib.pyplot figure.

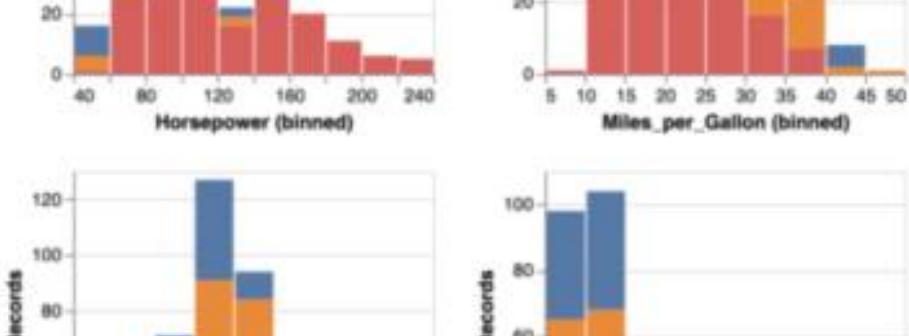
```
st.pyplot(my_mpl_figure)
```



## Altair

Display a chart using the Altair library.

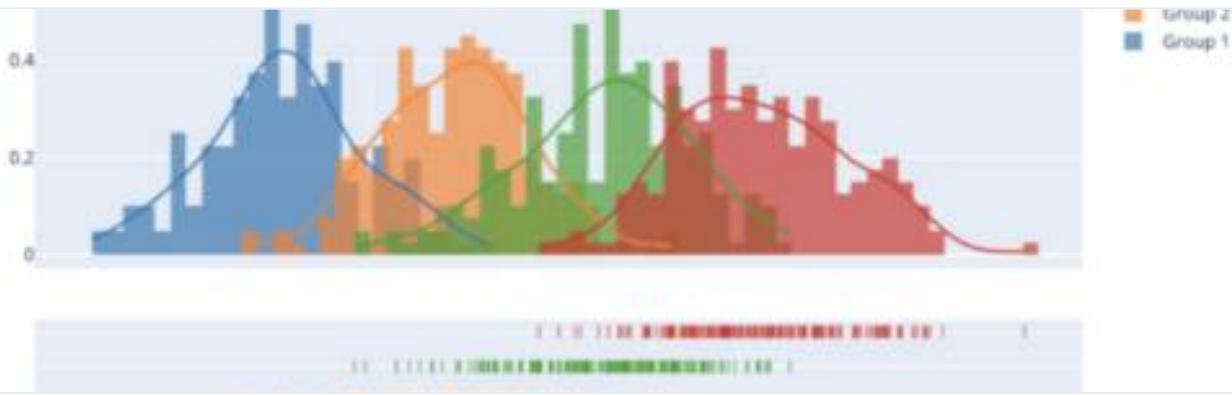
```
st.altair_chart(my_altair_chart)
```



## Vega-Lite

Display a chart using the Vega-Lite library.

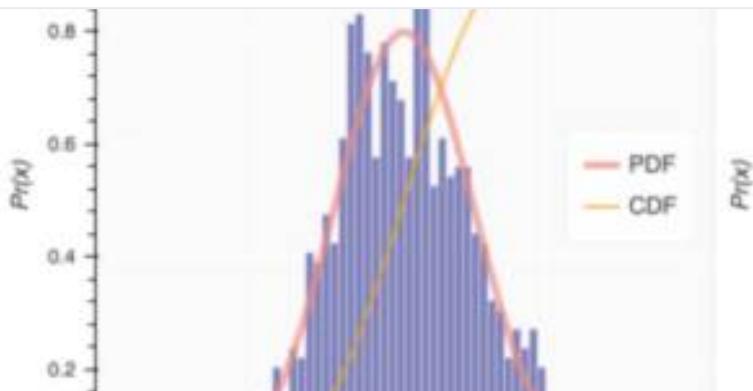
```
st.vega_lite_chart(my_vega_lite_chart)
```



## Plotly

Display an interactive Plotly chart.

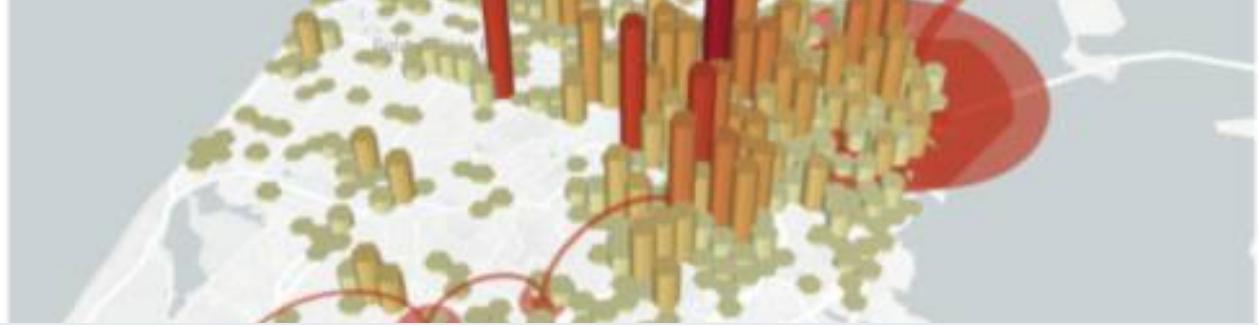
```
st.plotly_chart(my_plotly_chart)
```



## Bokeh

Display an interactive Bokeh chart.

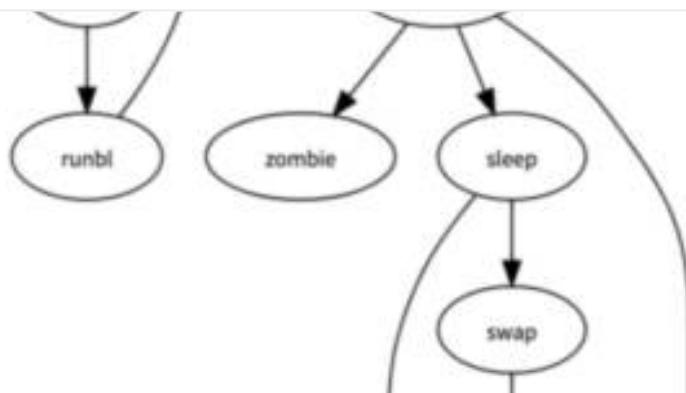
```
st.bokeh_chart(my_bokeh_chart)
```



## PyDeck

Display a chart using the PyDeck library.

```
st.pydeck_chart(my_pydeck_chart)
```



## GraphViz

Display a graph using the dagre-d3 library.

```
st.graphviz_chart(my_graphviz_spec)
```

## Input widgets



## Button

Display a button widget.

```
clicked = st.button("Click me")
```



## Download button

Display a download button widget.

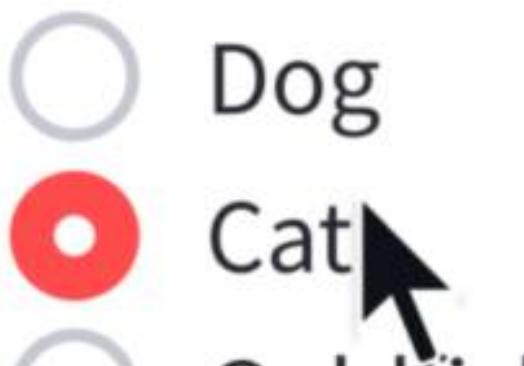
```
st.download_button("Download file", file)
```



## Checkbox

Display a checkbox widget.

```
selected = st.checkbox("I agree")
```



## Radio

Display a radio button widget.

```
choice = st.radio("Pick one", ["cats", "dogs"])
```

```
cats
```

cats

dogs



## Selectbox

Display a select widget.

```
choice = st.selectbox("Pick one", ["cats", "dogs"])
```

milk X

bananas X



apples

potatoes



## Multiselect

Display a multiselect widget. The multiselect widget starts as empty.

```
choices = st.multiselect("Buy", ["milk", "apples", "potatoes"])
```

## Pick a number

42

0

100



## Slider

Display a slider widget.

```
number = st.slider("Pick a number", 0, 100)
```

## Dominant color in image

## Dominant color in image



## Select-slider

Display a slider widget to select items from a list.

```
size = st.select_slider("Pick a size", ["S", "M", "L"])
```

Movie title

Life of Brian



## Text input

Display a single-line text input widget.

```
name = st.text_input("First name")
```

Number of days

28

- +



## Number input

Display a numeric input widget.

```
choice = st.number_input("Pick a number", 0, 10)
```

Text to analyze

It was the best of times, it was the worst of times, it was the age of wisdom,  
it was the age of foolishness, it was the epoch of belief, it was the epoch of  
incredulity, it was the season of Light as the season of Darkness, it was  
the spring of hope, it was the winter of despair. . .

## Text-area

Display a multi-line text input widget.

```
text = st.text_area("Text to translate")
```



## Date input

Display a date input widget.

```
date = st.date_input("Your birthday")
```

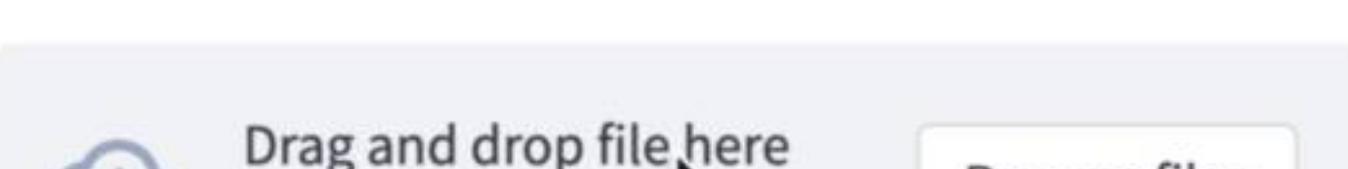


## Time input

Display a time input widget.

```
time = st.time_input("Meeting time")
```

## Choose a CSV file





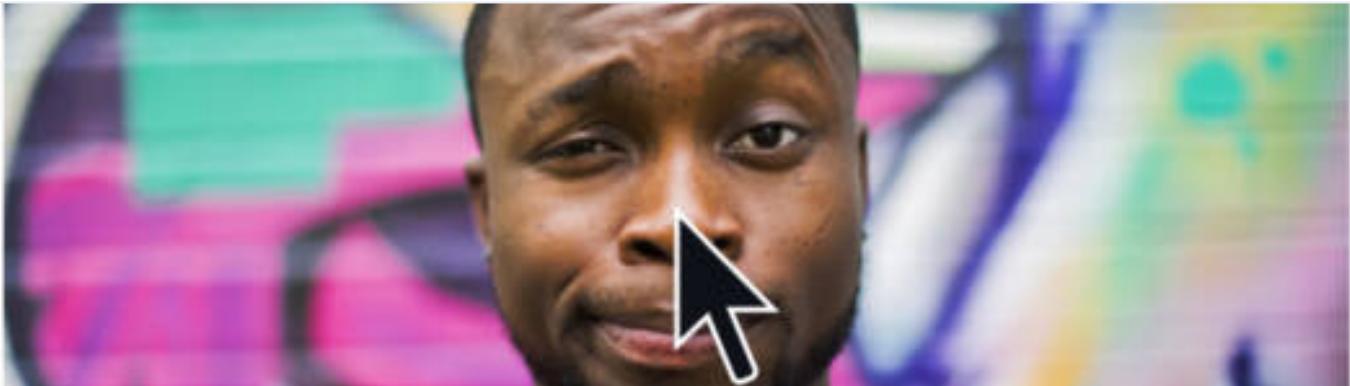
Limit 200MB per file

Browse files

## File Uploader

Display a file uploader widget.

```
data = st.file_uploader("Upload a CSV")
```



## Camera input

Display a widget that allows users to upload images directly from a camera.

```
image = st.camera_input("Take a picture")
```



## Color picker

Display a color picker widget.

```
color = st.color_picker("Pick a color")
```

## Media elements

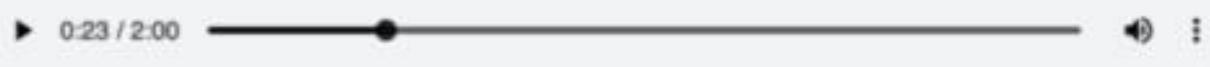




## Image

Display an image or list of images.

```
st.image(numpy_array)
st.image(image_bytes)
st.image(file)
st.image("https://example.com/myimage.jpg")
```



## Audio

Display an audio player.

```
st.audio(numpy_array)
st.audio(audio_bytes)
st.audio(file)
st.audio("https://example.com/myaudio.mp3", format="audio/mp3")
```



## Video

Display a video player.

```
st.video(numpy_array)  
st.video(video_bytes)  
st.video(file)  
st.video("https://example.com/myvideo.mp4", format="video/mp4")
```

## Layouts and containers

Adipiscing elit, duis tristique sollicitudin. Velit aliquet sagittis id consectetur purus ut faucibus pulvinar.

### Tincidunt lobortis

Feugiat vivamus at augue eget arcu dictum. Sed risus pretium quam vulputate. Cursus in hac habitasse platea dictumst. Aliquam ultrices sagittis orci a.

### Non diam phasellus vestibulum

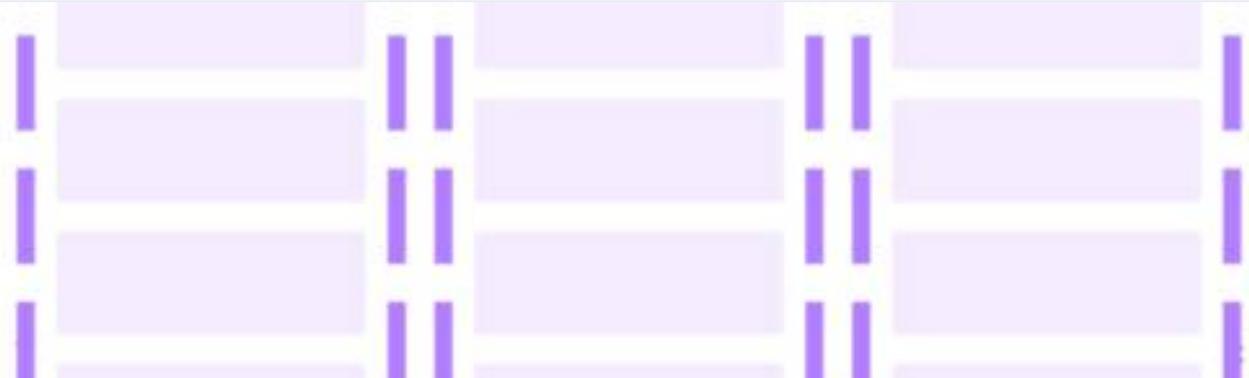
Vel quam elementum pulvinar etiam. Blandit volutpat maecenas volutpat blandit aliquam. Est sit amet facilisis magna etiam tempor orci eu lobortis.

[Read more >](#)

## Sidebar

Display items in a sidebar.

```
st.sidebar.write("This lives in the sidebar")  
st.sidebar.button("Click me!")
```



## Columns

Insert containers laid out as side-by-side columns.

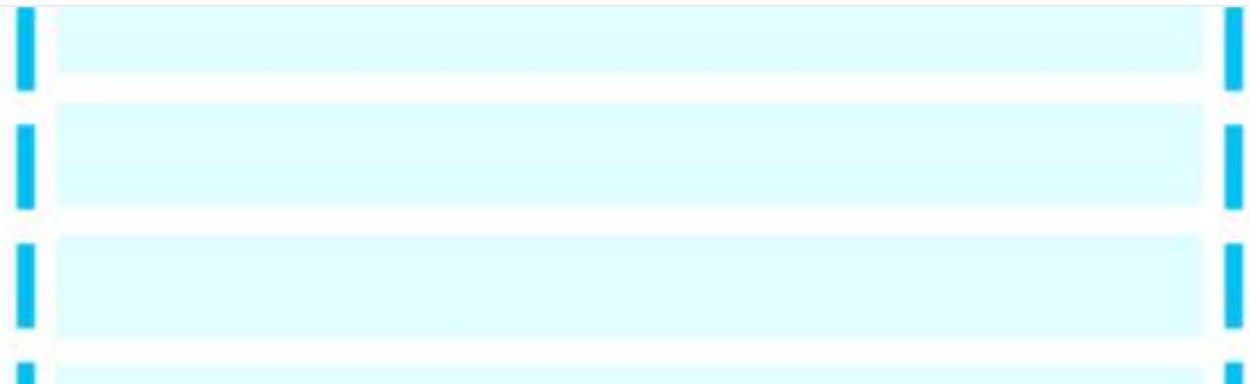
```
col1, col2 = st.columns(2)
col1.write("this is column 1")
col2.write("this is column 2")
```



## Expander

Insert a multi-element container that can be expanded/collapsed.

```
with st.expander("Open to see more"):
    st.write("This is more content")
```



## Container

Insert a multi-element container.

```
c = st.container()
st.write("This will show last")
c.write("This will show first")
c.write("This will show second")
```



## Empty

Insert a single-element container.

```
c = st.empty()  
st.write("This will show last")  
c.write("This will be replaced")  
c.write("This will show first")
```

## Display progress and status



## Progress bar

Display a progress bar.

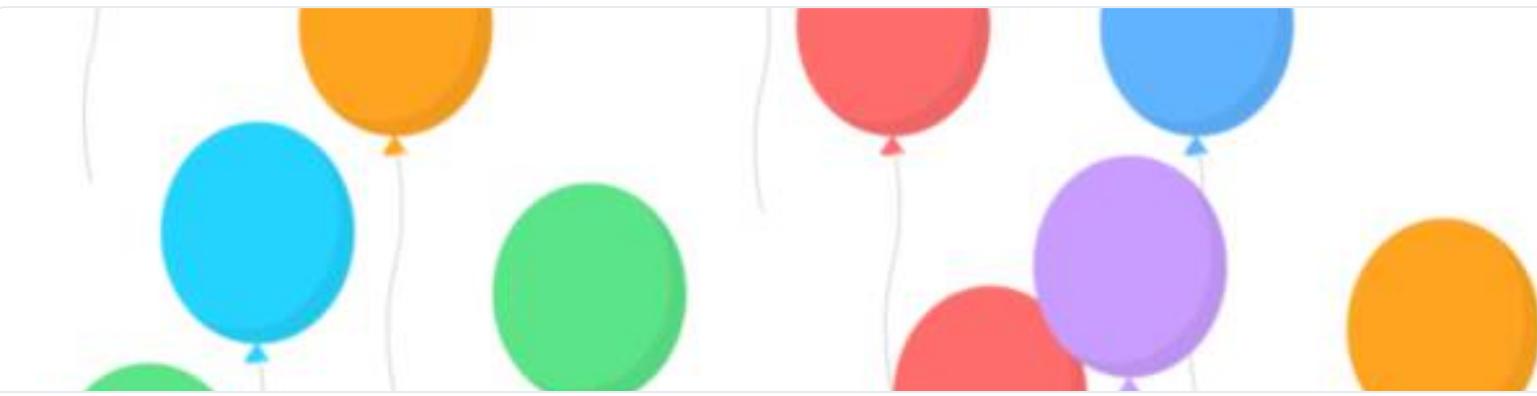
```
for i in range(101):  
    st.progress(i)  
    do_something_slow()
```



## Spinner

Temporarily displays a message while executing a block of code.

```
with st.spinner("Please wait..."):  
    do_something_slow()
```



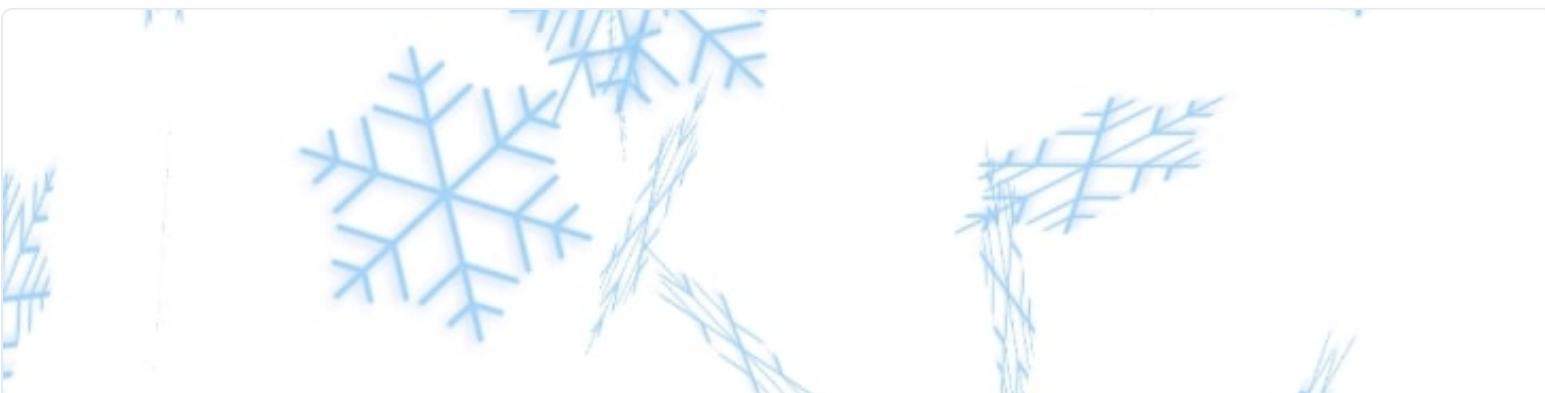
## Balloons

Display celebratory balloons!

```
do_something()
```

```
# Celebrate when all done!
```

```
st.balloons()
```



## Snowflakes

Display celebratory snowflakes!

```
do_something()
```

```
# Celebrate when all done!
```

```
st.snow()
```

Error running analysis

## Error box

Display error message.

```
st.error("We encountered an error")
```

Trimmed input to fit buffer

## Warning box

Display warning message.

```
st.warning("Unable to fetch image. Skipping...")
```

Last datapoint is for partial data

## Info box

Display an informational message.

```
st.info("Dataset is updated every day at midnight.")
```

Analysis finished in 3.14s

## Success box

Display a success message.

```
st.success("Match found!")
```

ValueError: operands could not be broadcast together with shapes (10,2) (2,10)

Traceback:

```
File "/Users/tvst/Projects/streamlit/streamlit/docs/api-examples-source/tit  
np.random.randn(10,2) * np.random.randn(2, 10)
```

## Exception output

Display an exception.

```
e = RuntimeError("This is an exception of type RuntimeError")  
st.exception(e)
```

## Control flow

### Forms

Create a form that batches elements together with a "Submit" button.

```
with st.form(key='my_form'):  
    username = st.text_input("Username")  
    password = st.text_input("Password")  
    st.form_submit_button("Login")
```

### Stop execution

Stops execution immediately.

```
st.stop()
```

## Rerun script

Refun the script immediately.

```
st.experimental_rerun()
```

# Utilities

## Set page title, favicon, and more

Configures the default settings of the page.

```
st.set_page_config(  
    title="My app",  
    favicon=":shark:",  
)
```

## Echo

Display some code on the app, then execute it. Useful for tutorials.

```
with st.echo():  
    st.write('This code will be printed')
```

## Get help

Display object's doc string, nicely formatted.

```
st.help(st.write)  
st.help(pd.DataFrame)
```

## st.experimental\_show

Write arguments and argument names to your app for debugging purposes.

```
df = pd.DataFrame({  
    'first column': [1, 2, 3, 4],  
    'second column': [10, 20, 30, 40],
```

```
)
```

## Get query paramters

Return the query parameters that are currently showing in the browser's URL bar.

```
st.experimental_get_query_params()
```

## Set query paramters

Set the query parameters that are shown in the browser's URL bar.

```
st.experimental_set_query_params(  
    show_map=True,  
    selected=["asia"]  
)
```

## Mutate charts

### Add rows

Append a dataframe to the bottom of the current one in certain elements, for optimized data updates.

```
element = st.line_chart(df)  
element.add_rows(df_with_extra_rows)
```

## State management

### Session state

Session state is a way to share variables between reruns, for each user session.

```
st.session_state['key'] = value
```

# Performance

## Caching

Function decorator to memoize function executions.

```
@st.cache(ttl=3600)  
def run_long_computation(arg1, arg2):  
    # Do stuff here  
    return computation_output
```

## Memo

Experimental function decorator to memoize function executions.

```
@st.experimental_memo  
def fetch_and_clean_data(url):  
    # Fetch data from URL here, and then clean it up.  
    return data
```

## Singleton

Experimental function decorator to store singleton objects.

```
@st.experimental_singleton  
def get_database_session(url):  
    # Create a database session object that points to the URL.  
    return session
```

## Clear memo

Clear all in-memory and on-disk memo caches.

```
@st.experimental_memo  
def fetch_and_clean_data(url):  
    # Fetch data from URL here, and then clean it up.  
    return data  
  
if st.checkbox("Clear All"):
```

## Clear singleton

Clear all singleton caches.

```
@st.experimental_singleton  
def get_database_session(url):  
    # Create a database session object that points to the URL.  
    return session  
  
if st.button("Clear All"):  
    # Clears all singleton caches:  
    st.experimental_singleton.clear()
```

## Was this page helpful?

 Yes     No

 [Suggest edits](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

[← Previous: Get started](#)

[Next: Write and magic →](#)



## Documentation

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [\*cloud\*](#)

### [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notes](#)*open in new*
- [Troubleshooting](#)

- [\*school\*](#)

### [Knowledge base](#)

- [Tutorials](#)  
*add*
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [\*Home\*](#)/
- [\*Knowledge base\*](#)/
- [\*Using Streamlit\*](#)/
- [\*Append data to a table or chart\*](#)

## **Append data to a table or chart**

*8*

In Streamlit, you can not only replace entire elements in your app, but also modify the data behind those elements. Here is how:

```
import numpy as np
import time

# Get some data.
data = np.random.randn(10, 2)

# Show the data as a chart.
chart = st.line_chart(data)

# Wait 1 second, so the change is clearer.
```

```
time.sleep(1)

# Grab some more data.
data2 = np.random.randn(10, 2)

# Append the new data to the existing chart.
chart.add_rows(data2)
```

Was this page helpful?

Yes  No

[edit](#)[Suggest edits](#)

[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

←[Previous: How to animate elements?](#)[Next: Batch elements and input widgets with st.form](#)→

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.

## Documentation

*search*

Search

- *description*

### Streamlit library

- Get started  
*add*
- API reference  
*add*
- Advanced features  
*add*
- Components  
*add*
- Changelog
- Cheat sheet

- *cloud*

### Streamlit Cloud

- Get started  
*add*
- Trust and Security
- Release notes*open in new*
- Troubleshooting

- *school*

### Knowledge base

- Tutorials  
*remove*
  - Connect to data sources  
*remove*
    - AWS S3
    - BigQuery
    - Snowflake
    - Microsoft SQL Server
    - Firebase*open in new*
    - MongoDB
    - MySQL
    - PostgreSQL
    - Tableau
    - Private Google Sheet
    - Public Google Sheet
    - TigerGraph
    - Deta Base
    - Supabase
    - Google Cloud Storage
  - Session State basics
- Using Streamlit
- Streamlit Components
- Installing dependencies
- Deployment issues

# Introduction

This guide explains how to securely access files on AWS S3 from Streamlit Cloud. It uses the [s3fs](#) library and Streamlit's [secrets management](#).

## Create an S3 bucket and add a file



If you already have a bucket that you want to use, feel free to [skip to the next step](#).

First, [sign up for AWS](#) or log in. Go to the [S3 console](#) and create a new bucket:

A screenshot of the AWS S3 Management Console. The browser title bar shows "S3 Management Console". The URL is "s3.console.aws.amazon.com/s3/home?region=us-east-1#buckets". The page header includes the AWS logo, "Services", and user info "Johannes Rieke". A sidebar on the left shows "Amazon S3". The main content area has a heading "Buckets (3)". Below it, a sub-header says "Buckets are containers for data stored in S3. [Learn more](#)". There are four buttons: "C" (Create), "Copy ARN", "Empty", and "Delete". To the right of these is a large orange "Create bucket" button, which is circled in red. Below these buttons is a search bar with placeholder text "Find buckets by name". At the bottom of the table area are navigation icons for "1" and "2" pages, and a settings gear icon. A table below the buttons lists three buckets: one row is visible with columns "Name", "AWS Region", "Access", and "Creation date". The "Name" column contains two circular icons. The "AWS Region" column contains "us-east-1". The "Access" column contains "Public". The "Creation date" column contains "2021-07-01". At the very bottom of the page are links for "Feedback", "English (US)", "Privacy Policy", "Terms of Use", and "Cookie preferences". The footer copyright notice is "© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved."

The screenshot shows the AWS S3 Management Console interface. At the top, there's a navigation bar with the AWS logo, a search bar, and user profile information for 'Johannes Rieke'. Below the navigation bar, the main content area has a left sidebar with a menu icon and the text 'Amazon S3 > Create bucket'. The main content area is titled 'Create bucket' and contains a sub-section titled 'General configuration'. It includes fields for 'Bucket name' (containing 'streamlitbucket'), 'AWS Region' (set to 'US East (N. Virginia) us-east-1'), and an optional 'Copy settings from existing bucket - *optional*' section. At the bottom of the page, there are links for 'Feedback', language selection ('English (US)'), and legal links ('Privacy Policy', 'Terms of Use', 'Cookie preferences'). A copyright notice at the very bottom states '© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.'

Navigate to the upload section of your new bucket:

S3 Management Console

s3.console.aws.amazon.com/s3/ho...

aws Services ▾

Buckets (4)

Buckets are containers for data stored in S3. [Learn more](#)

C Copy ARN Empty Delete Create bucket

Find buckets by name

< 1 > | ⚙

Name	AWS Region	Access	Creation date
[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]
[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]
streamlitbucket	US East (N. Virginia) us-east-1	Bucket and objects not public	April 14, 2021, 23:41:31 (UTC+02:00)

Feedback English (US) ▾

Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The screenshot shows the AWS S3 Management Console interface. At the top, there's a navigation bar with the AWS logo, a search bar, and user information for 'Johannes Rieke'. Below the navigation is a section titled 'Buckets (4)' with a sub-instruction 'Buckets are containers for data stored in S3.' followed by a 'Learn more' link. A toolbar below this includes buttons for 'Copy ARN', 'Empty', 'Delete', and a prominent orange 'Create bucket' button. A search bar labeled 'Find buckets by name' is also present. The main area displays a table of buckets. The first two rows have their details redacted. The third row contains a bucket named 'streamlitbucket', which is circled in red. The fourth row shows the full details: 'streamlitbucket' in 'us-east-1' region, with access set to 'Bucket and objects not public', and a creation date of April 14, 2021, at 23:41:31 UTC+02:00. At the bottom, there are links for 'Feedback', language selection ('English (US)'), and legal links ('Privacy Policy', 'Terms of Use', 'Cookie preferences'). A copyright notice for Amazon Web Services from 2008 to 2021 is also at the bottom.

The screenshot shows the AWS S3 Management Console interface. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, search icon, notifications, user 'Johannes Rieke', 'Global' dropdown, and 'Support' dropdown. Below the navigation bar, the path 'Amazon S3 > streamlitbucket' is shown. The main title 'streamlitbucket' is displayed. A horizontal menu bar below the title includes 'Objects' (which is orange and underlined), 'Properties', 'Permissions', 'Metrics', 'Management', and 'Access Points'. The 'Objects' section is active, showing 'Objects (0)'. A descriptive text block explains that objects are fundamental entities stored in Amazon S3 and provides a link to 'Amazon S3 inventory'. Below this are buttons for 'Delete', 'Actions', 'Create folder', and a prominent orange 'Upload' button, which is circled in red. There's also a search bar labeled 'Find objects by prefix' and a pagination area with page number '1'. At the bottom, there are links for 'Feedback', 'English (US)', 'Privacy Policy', 'Terms of Use', and 'Cookie preferences', along with a copyright notice: '© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.'

And upload the following CSV file, which contains some example data:

[myfile.csv](#)

## Create access keys

Go to the [AWS console](#), create access keys as shown below and copy the "Access Key ID" and "Secret Access Key":

AWS Management Console

console.aws.amazon.com/console/home?region=us-east-1#security\_credentials

aws Services Search Johannes Rieke N. Virginia Support

# AWS Management

**AWS services**

Recently visited services: IAM

All services:

- Compute**: EC2, Lightsail, Lambda, Batch, Elastic Beanstalk
- Machine Learning**: Amazon SageMaker, Amazon Augmented AI, Amazon CodeGuru, Amazon DevOps Guru, Amazon Comprehend

My Account: 415334732223  
My Organization  
My Service Quotas  
My Billing Dashboard  
**My Security Credentials** (circled)  
Sign Out

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

https://console.aws.amazon.com/iam/home?region=us-east-1#security\_credentials

IAM Management Console

console.aws.amazon.com/iam/home?region=us-east-1#security\_credentials

aws Services Search Johannes Rieke Global Support

Identity and Access

- Dashboard
- ▼ Access management
  - Groups
  - Users
  - Roles
  - Policies
  - Identity providers
  - Account settings
- ▼ Access reports
  - Access analyzer
  - Archive rules
  - Analyzers
  - Settings

[Feedback](#) English (US) ▾

▲ Password

▲ Multi-factor authentication (MFA)

▼ Access keys (access key ID and secret access key)

Use access keys to make programmatic calls to AWS from the AWS CLI, Tools for PowerShell, or AWS SDKs. You can have a maximum of two access keys (active or inactive) at a time.

For your protection, you should never share your secret keys with anyone. As a best practice, if you lose or forget your secret key, you cannot retrieve it. Instead, create a new access key and delete the old one.

Created	Access Key ID	Last Used

**Create New Access Key**

Root user access keys provide unrestricted access to your entire AWS account. If you want to give your application access to specific AWS services, consider creating a new IAM user with limited permissions and generating access keys for that user instead.

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

*star* **Tip**

Access keys created as a root user have wide-ranging permissions. In order to make your AWS account more secure, you should consider creating an IAM account with restricted permissions and using its access keys. More information [here](#).

## Add the key to your local app secrets

Your local Streamlit app will read secrets from a file `.streamlit/secrets.toml` in your app's root directory. Create this file if it doesn't exist yet and add the access key to it as shown below:

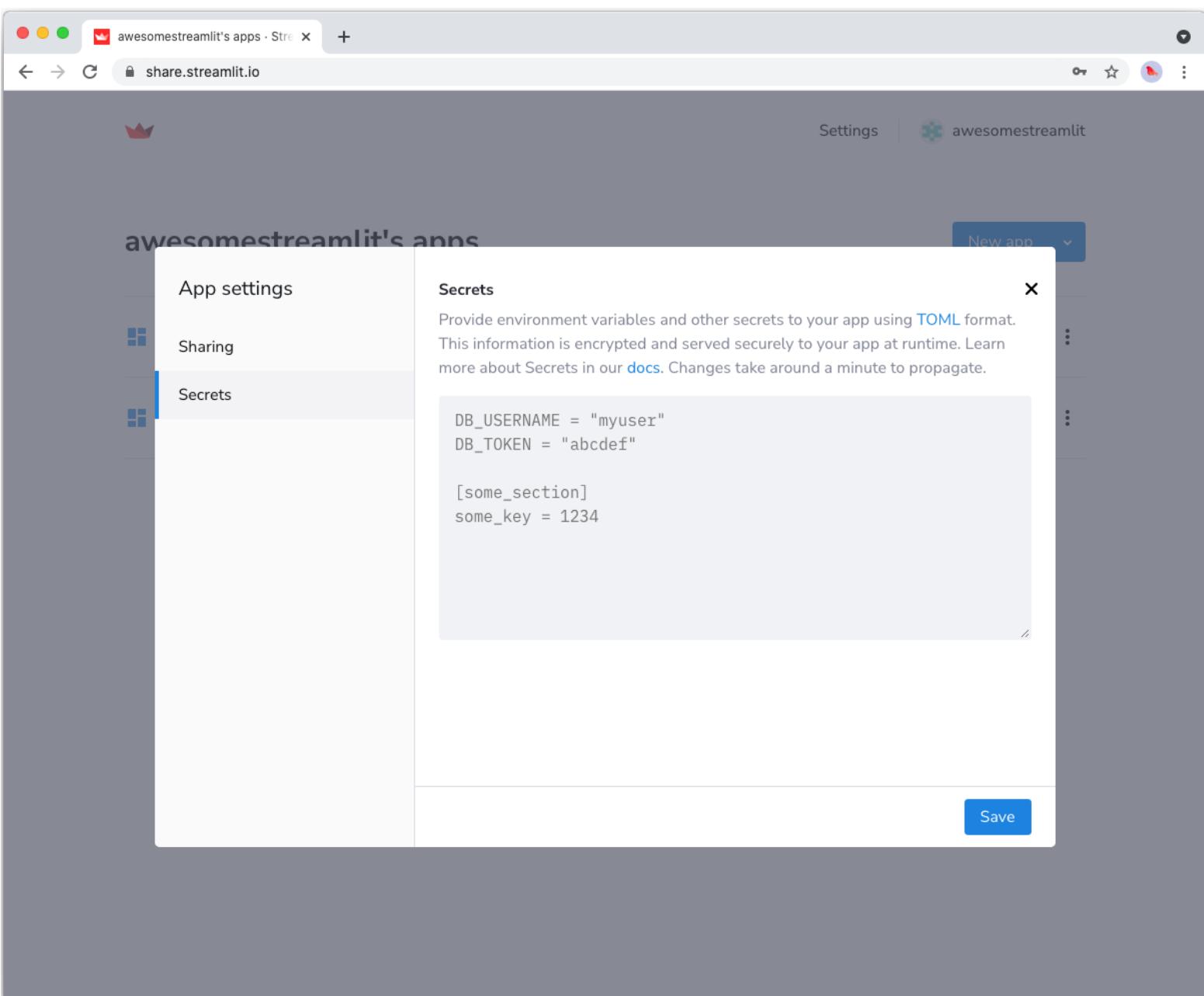
```
# .streamlit/secrets.toml
AWS_ACCESS_KEY_ID = "xxx"
AWS_SECRET_ACCESS_KEY = "xxx"
```

## *priority\_high* Important

Add this file to `.gitignore` and don't commit it to your Github repo!

## Copy your app secrets to the cloud

As the `secrets.toml` file above is not committed to Github, you need to pass its content to your deployed app (on Streamlit Cloud) separately. Go to the [app dashboard](#) and in the app's dropdown menu, click on **Edit Secrets**. Copy the content of `secrets.toml` into the text area. More information is available at [Secrets Management](#).



## Add s3fs to your requirements file

Add the [s3fs](#) package to your `requirements.txt` file, preferably pinning its version (replace `x.x.x` with the version you want installed):

```
# requirements.txt
s3fs==x.x.x
```

## Write your Streamlit app

Copy the code below to your Streamlit app and run it. Make sure to adapt the name of your bucket and file. Note that Streamlit automatically turns the access keys from your secrets file into environment variables, where `s3fs` searches for them by default.

```
# streamlit_app.py

import streamlit as st
import s3fs
import os

# Create connection object.
# `anon=False` means not anonymous, i.e. it uses access keys to pull data.
fs = s3fs.S3FileSystem(anon=False)

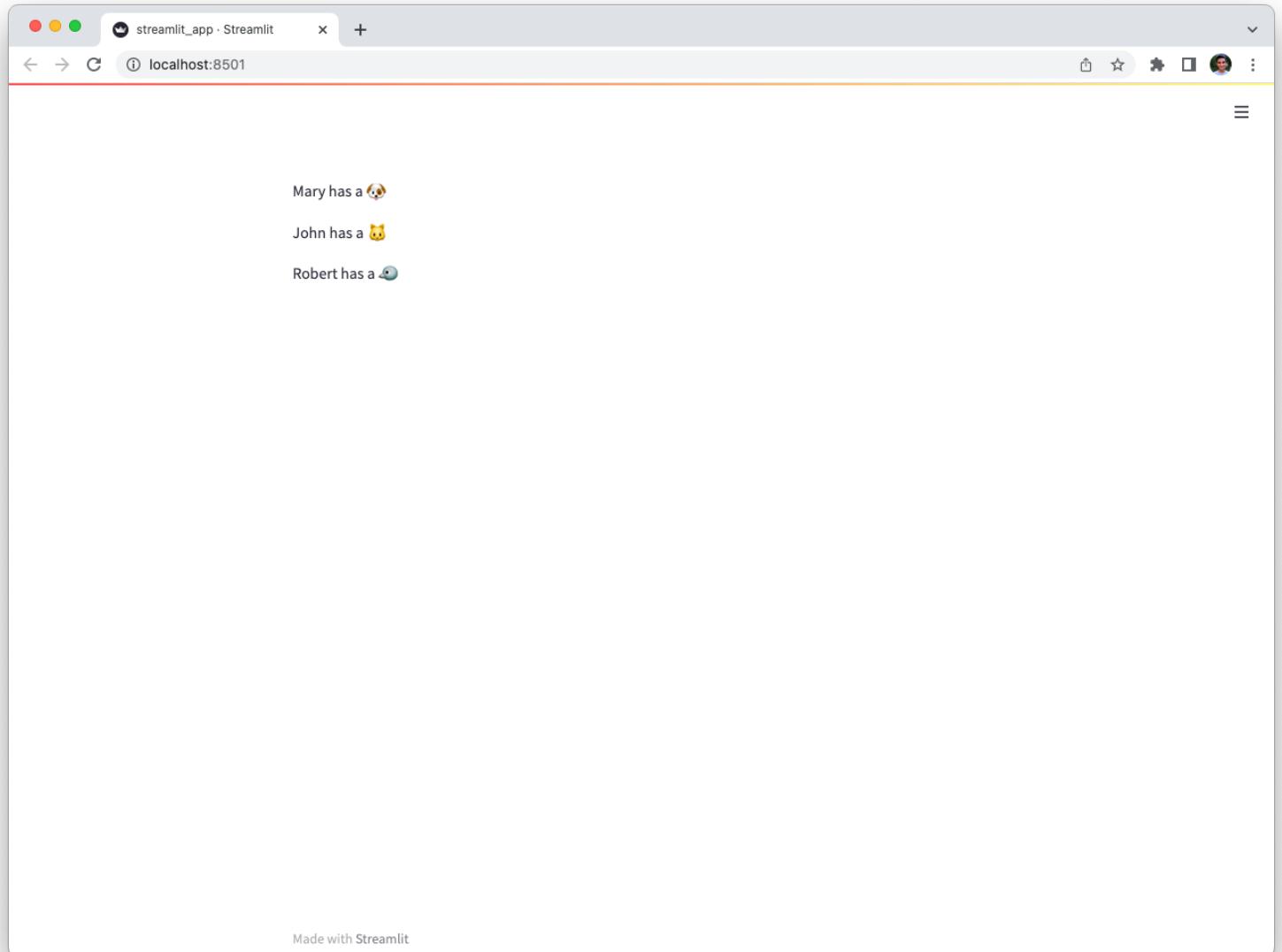
# Retrieve file contents.
# Uses st.experimental_memo to only rerun when the query changes or after 10 min.
@st.experimental_memo(ttl=600)
def read_file(filename):
    with fs.open(filename) as f:
        return f.read().decode("utf-8")

content = read_file("testbucket-jrieke/myfile.csv")

# Print results.
for line in content.strip().split("\n"):
    name, pet = line.split(",")
    st.write(f"{name} has a :{pet}:")
```

See `st.experimental_memo` above? Without it, Streamlit would run the query every time the app reruns (e.g. on a widget interaction). With `st.experimental_memo`, it only runs when the query changes or after 10 minutes (that's what `ttl` is for). Watch out: If your database updates more frequently, you should adapt `ttl` or remove caching so viewers always see the latest data. Read more about caching [here](#).

If everything worked out (and you used the example file given above), your app should look like this:



**Was this page helpful?**

*thumb\_up* Yes      *thumb\_down* No

*edit* [Suggest edits](#)

**Still have questions?**

*forum*

Our [forums](#) are full of helpful information and Streamlit experts.





# Batch elements and input widgets with st.form

Let's take a look at how to use `st.form` to batch elements and input widgets.

In Streamlit, every widget interaction causes a rerun of the app. However, there are times when you might want to interact with a couple of widgets and submit those interactions while triggering a single re-run of the app.

Using `st.form` you can batch input widgets together and along with `st.form_submit_button` submit the state inside these widgets with the click of a single button.

```
# Forms can be declared using the 'with' syntax
with st.form(key='my_form'):
    text_input = st.text_input(label='Enter your name')
    submit_button = st.form_submit_button(label='Submit')
```

```
# Alternative syntax, declare a form and use the returned object
form = st.form(key='my_form')
form.text_input(label='Enter some text')
submit_button = form.form_submit_button(label='Submit')
```

```
# st.form_submit_button returns True upon form submit
if submit_button:
    st.write(f'Hello {name}!')
```

Forms can appear anywhere in your app (sidebar, columns etc), but there are some **constraints**:

- A form cannot have interdependent widgets, i.e. the *output* of `widget1` cannot be the *input* to `widget2` inside a form.

- By design, interacting with widgets inside `st.form` does not trigger a re-run. Because of this reason, `st.button` cannot be declared inside `st.form`.
- `st.form` cannot be embedded inside another `st.form`.
- Forms must have an associated `st.form_submit_button`. Clicking this button triggers a re-run. Streamlit throws an error if a form does not have an associated `st.form_submit_button`.

## Was this page helpful?

 Yes     No

 [Suggest edits](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

[← Previous: Append data to a table or chart](#)

[Next: Caching issues →](#)

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



## Connect Streamlit to Google BigQuery

### Introduction

This guide explains how to securely access a BigQuery database from Streamlit Cloud. It uses the [google-cloud-bigquery](#) library and Streamlit's [secrets management](#).

### Create a BigQuery database



If you already have a database that you want to use, feel free to [skip to the next step](#).

For this example, we will use one of the [sample datasets](#) from BigQuery (namely the `shakespeare` table). If you want to create a new dataset instead, follow [Google's quickstart guide](#).

### Enable the BigQuery API

Programmatic access to BigQuery is controlled through [Google Cloud Platform](#). Create an account or sign in and head over to the [APIs & Services dashboard](#) (select or create a project if asked). As shown below, search for the BigQuery API and enable it:

RPI APIs & Services – APIs & Services X +

← → C 🔒 console.cloud.google.com □ ⚙ ☆

☰ Google Cloud Platform My First Project 🔍 🎁 📈 ? 2 :

API APIs & Services **+ ENABLE APIs AND SERVICES**

🕒 1 hour 6 hours 12 hours 1 day 2 days 4 days 7 days 14 days 30 days

Traffic

0.04/s  
0.03/s  
0.02/s  
0.01/s

Apr 11 Apr 18 Apr 25 May 02 0

Errors

100%

The screenshot shows the Google Cloud Platform APIs & Services page. At the top, there's a navigation bar with tabs like 'My First Project', a search icon, and a notifications icon showing '2'. Below the navigation is a blue header bar with the title 'APIs & Services' and a prominent red-outlined button labeled '+ ENABLE APIs AND SERVICES'. To the left of the main content area is a sidebar with various icons for filtering and viewing data. The main content area has two sections: 'Traffic' and 'Errors'. The 'Traffic' section displays a line chart with four data points: 0.04/s, 0.03/s, 0.02/s, and 0.01/s. The x-axis shows dates from April 11 to May 02. The 'Errors' section shows a single data point of 100%. Both sections have download icons at the bottom right.

RPI API Library - My First Project - +

console.cloud.google.com

Google Cloud Platform My First Project

Search BigQuery

Filter by 9 results

CATEGORY

- Analytics (1)
- Big data (6)
- Developer tools (1)
- Google Cloud APIs (2)
- Healthcare (1)
- Other (2)

PRICE

- Free (1)
- Paid (2)

**BigQuery API**  
Google  
A data platform for customers to create, manage, share and query data.

**BigQuery Connection API**  
Google  
Allows users to manage BigQuery connections to external data sources.

**BigQuery Data Transfer API**  
Google  
Transfers data from partner SaaS applications to Google BigQuery on a scheduled basis.

**BigQuery Reservation API**

RPI BigQuery API – APIs & Services - +

console.cloud.google.com

Google Cloud Platform My First Project

BigQuery API

Google

A data platform for customers to create, manage, share and query data.

**ENABLE** TRY THIS API ↗

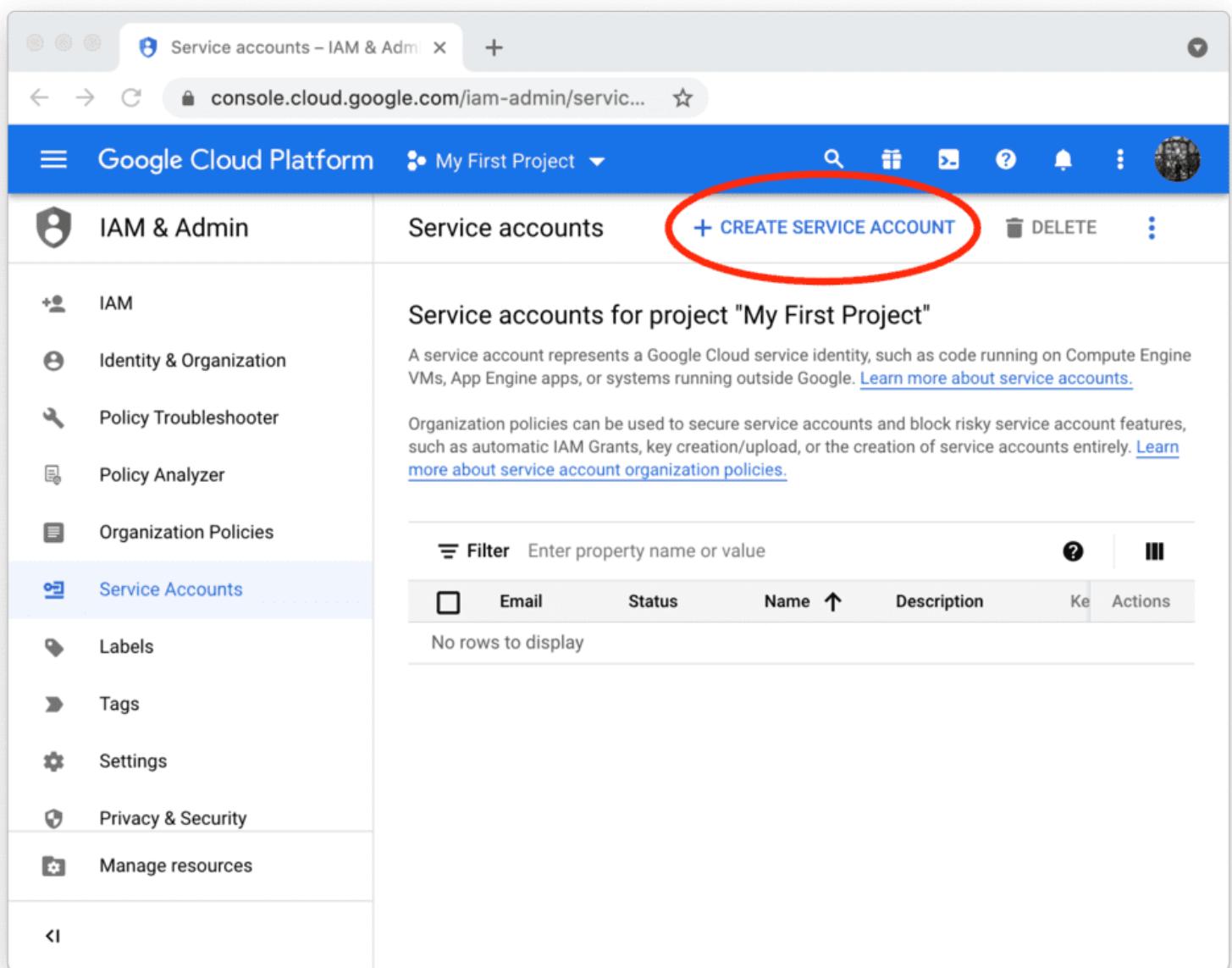
OVERVIEW DOCUMENTATION

## Overview

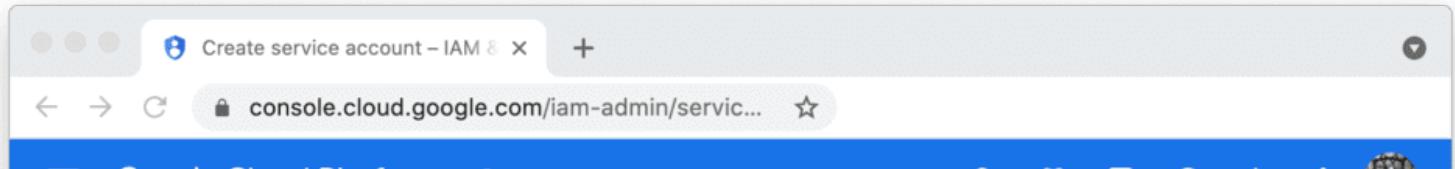
A data platform for customers to create, manage, share and query data.

# Create a service account & key file

To use the BigQuery API from Streamlit Cloud, you need a Google Cloud Platform service account (a special account type for programmatic data access). Go to the [Service Accounts](#) page and create an account with the **Viewer** permission (this will let the account access data but not change it):



The screenshot shows the Google Cloud Platform IAM & Admin Service Accounts page. The left sidebar has a 'Service Accounts' section highlighted. The main area shows a table with one row, indicating 'No rows to display'. A red circle highlights the '+ CREATE SERVICE ACCOUNT' button at the top right of the main content area.



The screenshot shows the 'Create service account - IAM' dialog box. It includes fields for 'Email', 'Status', 'Name' (with an upward arrow), 'Description', 'Key type', and 'Actions'. Below these fields, there is a note: 'This service account does not have any keys or credentials yet. You can add them later.' At the bottom, there is a 'Create' button.



IAM &amp; Admin



IAM



Identity &amp; Organization



Policy Troubleshooter



Policy Analyzer



Organization Policies



Service Accounts



Labels



Tags



Settings



Privacy &amp; Security



Identity-Aware Proxy



## Create service account

## ① Service account details

Service account name

service-acc

Display name for this service account

Service acco...

service-acc @striped-fulcrum-282122.iam.gserviceaccount.com

Service account description

Describe what this service account will do

CREATE

## ② Grant this service account access to project (optional)

## ③ Grant users access to this service account (optional)

The screenshot shows the Google Cloud Platform IAM & Admin service account creation interface. On the left, a sidebar lists various IAM-related options: IAM, Identity & Organization, Policy Troubleshooter, Policy Analyzer, Organization Policies, Service Accounts (which is selected and highlighted in blue), Labels, Tags, Settings, Privacy & Security, and Identity-Aware Proxy. The main panel is titled "Create service account" and contains two main sections: "Service account details" (marked with a checkmark) and "Grant this service account access to project (optional)". In the optional section, a "Role" dropdown is set to "Viewer", and a "Condition" link is available. A note below states: "Grant this service account access to My First Project so that it has permission to complete specific actions on the resources in your project." A "CONTINUE" button is present. At the bottom, there are "DONE" and "CANCEL" buttons.



If the button **CREATE SERVICE ACCOUNT** is gray, you don't have the correct permissions. Ask the admin of your Google Cloud project for help.

After clicking **DONE**, you should be back on the service accounts overview. Create a JSON key file for the new account and download it:

Service accounts – IAM & Admin

console.cloud.google.com/iam-admin/serviceaccounts

Google Cloud Platform My First Project

IAM & Admin Service accounts + CREATE SERVICE ACCOUNT DELETE

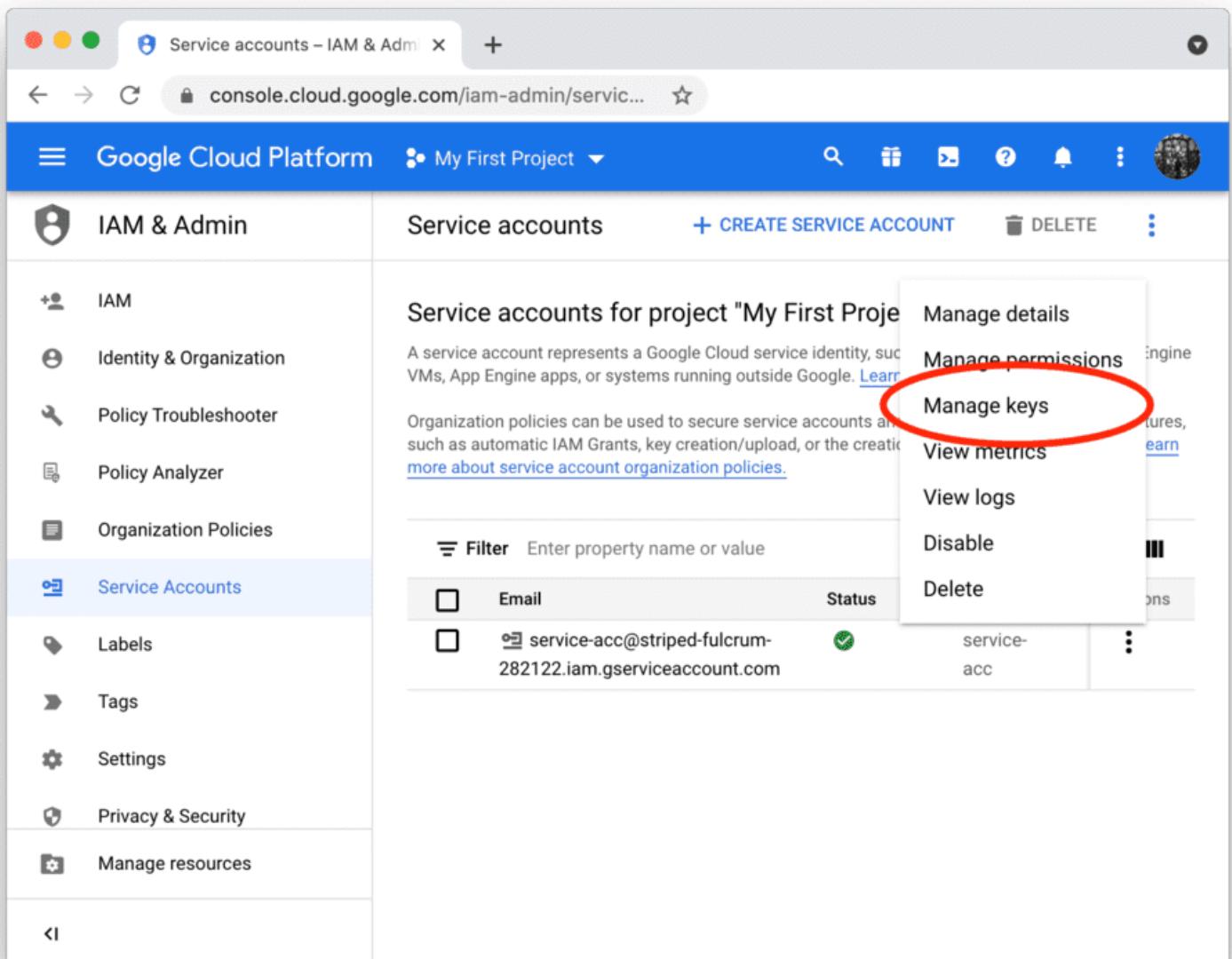
IAM Identity & Organization Policy Troubleshooter Policy Analyzer Organization Policies Service Accounts Labels Tags Settings Privacy & Security Manage resources

Service accounts for project "My First Proj" Manage details Manage permissions Manage keys View metrics View logs Disable Delete

Filter Enter property name or value

Email	Status
service-acc@striped-fulcrum-282122.iam.gserviceaccount.com	✓

Manage keys (circled)



service-acc – IAM & Admin

console.cloud.google.com/iam-admin/serviceaccounts

Google Cloud Platform My First Project

IAM & Admin service-acc DETAILS PERMISSIONS KEYS METRICS LOGS

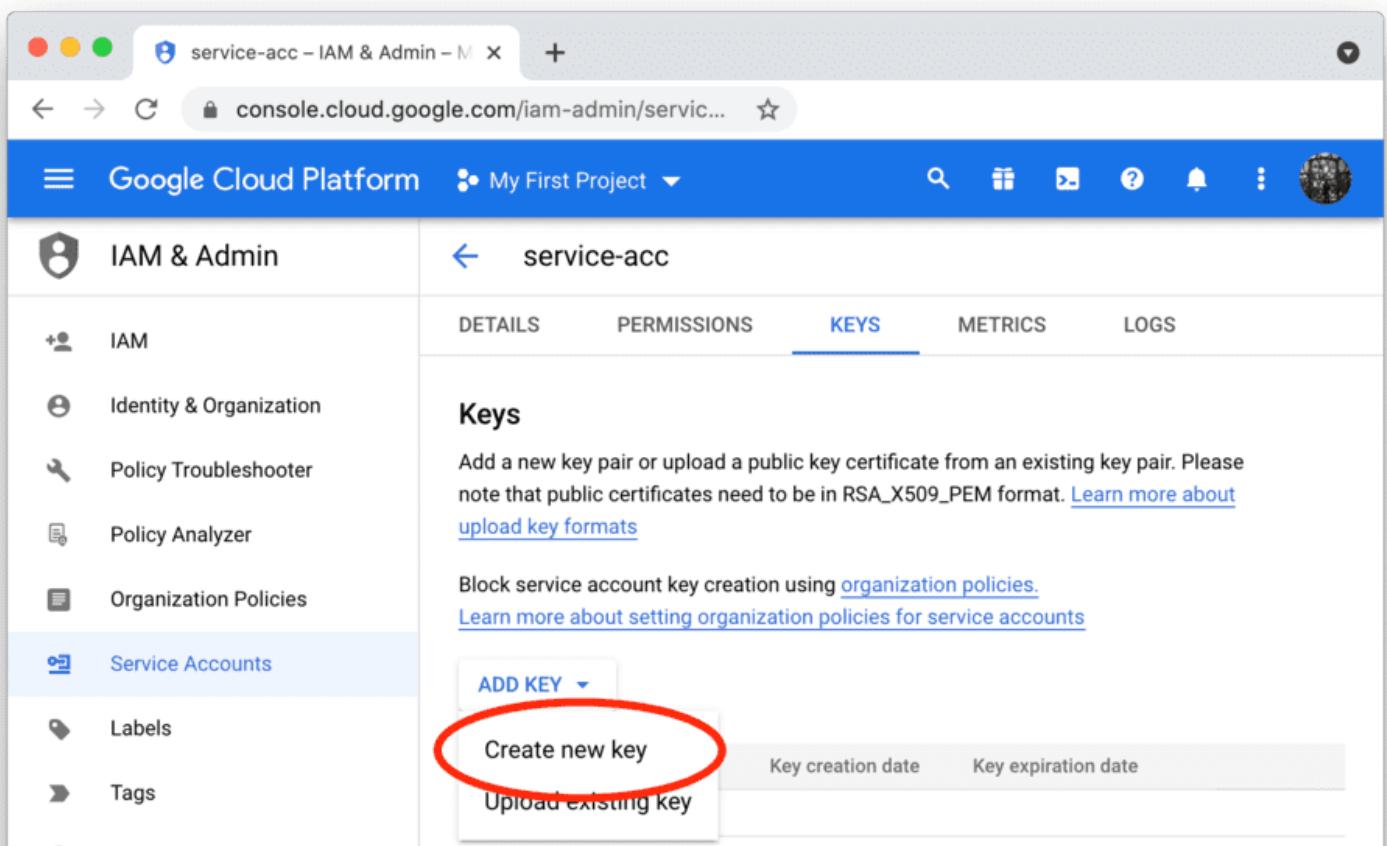
Keys

Add a new key pair or upload a public key certificate from an existing key pair. Please note that public certificates need to be in RSA\_X509\_PEM format. [Learn more about upload key formats](#)

Block service account key creation using [organization policies](#). [Learn more about setting organization policies for service accounts](#)

ADD KEY ▾ Create new key (circled) Upload existing key

Key creation date	Key expiration date
-------------------	---------------------



- Privacy & Security
- Identity-Aware Proxy

<

The screenshot shows a modal dialog box titled "Create private key for 'service-acc'". The dialog contains the following information:

- Description: "Downloads a file that contains the private key. Store the file securely because this key can't be recovered if lost."
- Key type:
  - JSON (Recommended)
  - P12 (For backward compatibility with code using the P12 format)
- Buttons: "CANCEL" and "CREATE"

The background shows the Google Cloud Platform interface with the "IAM & Admin" service selected. The sidebar includes options like IAM, Identity & Organization, Policy Troubleshooter, Policy Analyzer, Organization Policy, Service Accounts, Labels, Tags, Settings, Privacy & Security, and Identity-Aware Proxy.

## Add the key file to your local app secrets

Your local Streamlit app will read secrets from a file `.streamlit/secrets.toml` in your app's root directory. Create this file if it doesn't exist yet and add the content of the key file you just downloaded to it as shown below:

```
# .streamlit/secrets.toml

[gcp_service_account]
type = "service_account"
```



```
project_id = "xxx"
private_key_id = "xxx"
private_key = "xxx"
client_email = "xxx"
client_id = "xxx"
auth_uri = "https://accounts.google.com/o/oauth2/auth"
token_uri = "https://oauth2.googleapis.com/token"
auth_provider_x509_cert_url = "https://www.googleapis.com/oauth2/v1/certs"
client_x509_cert_url = "xxx"
```

*priority\_high***Important**

Add this file to `.gitignore` and don't commit it to your Github repo!

## Copy your app secrets to the cloud

As the `secrets.toml` file above is not committed to Github, you need to pass its content to your deployed app (on Streamlit Cloud) separately. Go to the [app dashboard](#) and in the app's dropdown menu, click on **Edit Secrets**. Copy the content of `secrets.toml` into the text area. More information is available at [Secrets Management](#).

The screenshot shows a Streamlit app settings page titled "awesomestreamlit's apps". On the left, there's a sidebar with "App settings", "Sharing", and "Secrets" options. The "Secrets" option is selected and highlighted with a blue border. A modal window titled "Secrets" is open, containing instructions about providing environment variables and secrets using TOML format. It includes a code editor with the following TOML content:

```
DB_USERNAME = "myuser"  
DB_TOKEN = "abcdef"  
  
[some_section]  
some_key = 1234
```

A "Save" button is located at the bottom right of the modal.

## Add google-cloud-bigquery to your requirements file

Add the [google-cloud-bigquery](#) package to your `requirements.txt` file, preferably pinning its version (replace `x.x.x` with the version want installed):

```
# requirements.txt  
google-cloud-bigquery==x.x.x
```

## Write your Streamlit app

Copy the code below to your Streamlit app and run it. Make sure to adapt the query if you don't use the sample table.



```
# streamlit_app.py

import streamlit as st
from google.oauth2 import service_account
from google.cloud import bigquery

# Create API client.
credentials = service_account.Credentials.from_service_account_info(
    st.secrets["gcp_service_account"])
)
client = bigquery.Client(credentials=credentials)

# Perform query.
# Uses st.experimental_memo to only rerun when the query changes or after 10 min.
@st.experimental_memo(ttl=600)
def run_query(query):
    query_job = client.query(query)
    rows_raw = query_job.result()
    # Convert to list of dicts. Required for st.experimental_memo to hash the return value.
    rows = [dict(row) for row in rows_raw]
    return rows

rows = run_query("SELECT word FROM `bigquery-public-data.samples.shakespeare` LIMIT 10")

# Print results.
st.write("Some wise words from Shakespeare:")
for row in rows:
    st.write("👉 " + row['word'])
```

See `st.experimental_memo` above? Without it, Streamlit would run the query every time the app reruns (e.g. on a widget interaction). With `st.experimental_memo`, it only runs when the query changes or after 10 minutes (that's what `ttl` is for). Watch out: If your database updates more frequently, you should adapt `ttl` or remove caching so viewers always see the latest data. Read more about caching [here](#).

Alternatively, you can use pandas to read from BigQuery right into a dataframe! Follow all the above steps, install the [pandas-gbq](#) library (don't forget to add it to `requirements.txt!`), and call `pandas.read_gbq(query, credentials=credentials)`. More info [in the pandas docs](#).

If everything worked out (and you used the sample table), your app should look like this:

Some wise words from Shakespeare:

- 👉 LVII
- 👉 augurs
- 👉 dimm'd
- 👉 plagues
- 👉 treason
- 👉 surmise
- 👉 heed
- 👉 Unthrifty
- 👉 quality
- 👉 wherever

**Was this page helpful?**

*thumb\_up* Yes      *thumb\_down* No

*edit* [Suggest edits](#)

**Still have questions?**

*forum*

Our [forums](#) are full of helpful information and Streamlit experts.

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.

## Documentation

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [\*cloud\*](#)

### [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notes open in new](#)
- [Troubleshooting](#)

- [\*school\*](#)

### [Knowledge base](#)

- [Tutorials](#)  
*add*
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [\*Home\*](#)/
- [\*Knowledge base\*](#)/
- [\*Using Streamlit\*](#)/
- [\*Caching issues\*](#)

## **Caching issues**

*8*

While developing an app, if you see an error or warning that stems from a cached function, it's probably related to the hashing procedure described in the [Optimize performance with `st.cache`](#). In this article, we'll provide solutions to common issues encountered when using caching. If you have an issue that's not covered in this article, please let us know in the [community forum](#).

## **How to debug a cached function that isn't executing**

*8*

If you believe your cached function isn't executing even though its inputs are a "Cache miss", you can debug using [`st.write`](#) statements inside and outside of your function like this:

```
@st.cache
def my_cached_func(a, b):
    st.write("Cache miss: my_cached_func(", a, ", ", b, ") ran")
    ...
    st.write("Calling my_cached_func(", a, ", ", b, ")")
    my_cached_func(2, 21)
```

## How to fix an UnhashableTypeError

Streamlit raises this error whenever it encounters a type it doesn't know how to hash. This could be either when hashing the inputs to generate the cache key or when hashing the output to verify whether it changed. To address it, you'll need to help Streamlit understand how to hash that type by using the `hash_funcs` argument:

```
@st.cache(hash_funcs={FooType: hash_foo_type})
def my_cached_func(a, b):
    ...
    ...
```

Here, `FooType` is the type Streamlit was unable to hash, and `hash_foo_type` is a function that can be used to properly hash `FooType` objects.

For example, if you'd like to make Streamlit ignore a specific type of object when hashing, you can pass a constant function to `hash_funcs`, like this:

```
@st.cache(hash_funcs={FooType: lambda _: None})
def my_cached_func(a, b):
    ...
    ...
```

For more information, see [Improve app performance](#).

# How to fix the Cached Object Mutated warning

8

By default Streamlit expects its cached values to be treated as immutable -- that cached objects remain constant. You received this warning if your code modified a cached object (see [Example 5 in Caching](#)). When this happens, you have a few options:

1. If you don't understand why you're seeing this error, it's very likely that you didn't mean to mutate the cached value in the first place. So you should either:
    - **Preferred:** rewrite your code to remove that mutation
    - Clone the output of the cached function before mutating it. For example:

```
import copy  
cloned_output = copy.deepcopy(my_cached_function(...))
```

2. If you wanted to allow the cached object to mutate, you can disable this check by setting `allow_output_mutation=True` like this:

```
@st.cache(allow_output_mutation=True)
def my_cached_func(...):
    ...

```

For examples, see [Advanced caching](#)

### *push pin*

## Note

If your function returns multiple objects and you only want to allow a subset of them to mutate between runs, you can do that with the `hash_funcs` option.

3. If Streamlit is incorrectly hashing the cached object, you can override this by using `hash_funcs`. For example, if your function returns an object of type `FooType` then you could write:

```
@st.cache(hash_funcs={FooType: hash_func_for_foo_type})  
def my_cached_func(...):  
    ...
```

For more information, see [Optimize performance with `st.cache`](#).

By the way, the scenario above is fairly unlikely — unless `FooType` does something particularly tricky internally. This is the case with some `SpaCY` objects, which can automatically mutate behind the scenes for better performance, while keeping their semantics constant. That means Streamlit will correctly detect a mutation in the object's internal structure, even though semantically that mutation makes no difference.

## If all else fails

8

If the proposed fixes above don't work for you, or if you have an idea on how to further improve [`st.cache`](#) -- let us know by asking questions in the [community forum](#), [filing a bug](#), or [submitting a feature request](#). We love hearing back from the community!

Was this page helpful?

`thumb_up` Yes    `thumb_down` No  
[edit](#) [Suggest edits](#)  
[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: Batch elements and input widgets with `st.form`](#) [Next: How do I create an anchor link? →](#)

---

[Home](#) [Contact Us](#) [Community](#)



Copyright © 2022, Streamlit Inc.



# Optimize performance with st.cache

## ! Important

We're developing new cache primitives that are easier to use and much faster than `@st.cache`. To learn more, read [Experimental cache primitives](#).

Streamlit provides a caching mechanism that allows your app to stay performant even when loading data from the web, manipulating large datasets, or performing expensive computations. This is done with the `@st.cache` decorator.

When you mark a function with the `@st.cache` decorator, it tells Streamlit that whenever the function is called it needs to check a few things:

1. The input parameters that you called the function with
2. The value of any external variable used in the function
3. The body of the function
4. The body of any function used inside the cached function

If this is the first time Streamlit has seen these four components with these exact values and in this exact combination and order, it runs the function and stores the result in a local cache. Then, next time the cached function is called, if none of these components changed, Streamlit will just skip executing the function altogether and, instead, return the output previously stored in the cache.

The way Streamlit keeps track of changes in these components is through hashing. Think of the cache as an in-memory key-value store, where the key is a hash of all of the above and the value is the actual output object passed by reference.

Finally, `@st.cache` supports arguments to configure the cache's behavior. You can find more information on those in our [API reference](#).

Let's take a look at a few examples that illustrate how caching works in a Streamlit app.

## Example 1: Basic usage

For starters, let's take a look at a sample app that has a function that performs an expensive, long-running computation. Without caching, this function is rerun each time the app is refreshed, leading to a poor user experience. Copy this code into a new app and try it out yourself:

```
import streamlit as st
import time

def expensive_computation(a, b):
    time.sleep(2) # 🚨 This makes the function take 2s to run
    return a * b

a = 2
b = 21
res = expensive_computation(a, b)

st.write("Result:", res)
```

Try pressing **R** to rerun the app, and notice how long it takes for the result to show up. This is because `expensive_computation(a, b)` is being re-executed every time the app runs. This isn't a great experience.

Let's add the `@st.cache` decorator:

```
import streamlit as st
import time

@st.cache # 🚨 Added this
def expensive_computation(a, b):
    time.sleep(2) # This makes the function take 2s to run
    return a * b

a = 2
b = 21
res = expensive_computation(a, b)

st.write("Result:", res)
```

Now run the app again and you'll notice that it is much faster every time you press R to rerun. To understand what is happening, let's add an st.write inside the function:

```
import streamlit as st
import time

@st.cache(suppress_st_warning=True) # 🤝 Changed this
def expensive_computation(a, b):
    # 🚧 Added this
    st.write("Cache miss: expensive_computation(", a, ", ", b, ") ran")
    time.sleep(2) # This makes the function take 2s to run
    return a * b

a = 2
b = 21
res = expensive_computation(a, b)

st.write("Result:", res)
```

Now when you rerun the app the text "Cache miss" appears on the first run, but not on any subsequent runs. That's because the cached function is only being executed once, and every time after that you're actually hitting the cache.

### ↗ Note

You may have noticed that we've added the `suppress_st_warning` keyword to the `@st.cache` decorators. That's because the cached function above uses a Streamlit command itself (`st.write` in this case), and when Streamlit sees that, it shows a warning that your command will only execute when you get a cache miss. More often than not, when you see that warning it's because there's a bug in your code. However, in our case we're using the `st.write` command to demonstrate when the cache is being missed, so the behavior Streamlit is warning us about is exactly what we want. As a result, we are passing in `suppress_st_warning=True` to turn that warning off.

## Example 2: When the function arguments change

Without stopping the previous app server, let's change one of the arguments to our cached function:

```

import streamlit as st
import time

@st.cache(suppress_st_warning=True)
def expensive_computation(a, b):
    st.write("Cache miss: expensive_computation()", a, ",", b, " ran")
    time.sleep(2) # This makes the function take 2s to run
    return a * b

a = 2
b = 210 # 🤝 Changed this
res = expensive_computation(a, b)

st.write("Result:", res)

```

Now the first time you rerun the app it's a cache miss. This is evidenced by the "Cache miss" text showing up and the app taking 2s to finish running. After that, if you press **R** to rerun, it's always a cache hit. That is, no such text shows up and the app is fast again.

This is because Streamlit notices whenever the arguments **a** and **b** change and determines whether the function should be re-executed and re-cached.

## Example 3: When the function body changes

Without stopping and restarting your Streamlit server, let's remove the widget from our app and modify the function's code by adding a **+ 1** to the return value.

```

import streamlit as st
import time

@st.cache(suppress_st_warning=True)
def expensive_computation(a, b):
    st.write("Cache miss: expensive_computation()", a, ",", b, " ran")
    time.sleep(2) # This makes the function take 2s to run
    return a * b + 1 # 🤝 Added a +1 at the end here

a = 2
b = 210
res = expensive_computation(a, b)

```

```
st.write("Result:", res)
```

The first run is a "Cache miss", but when you press **R** each subsequent run is a cache hit. This is because on first run, Streamlit detected that the function body changed, reran the function, and put the result in the cache.

### ★ Tip

If you change the function back the result will already be in the Streamlit cache from a previous run. Try it out!

## Example 4: When an inner function changes

Let's make our cached function depend on another function internally:

```
import streamlit as st
import time

def inner_func(a, b):
    st.write("inner_func()", a, ",", b, ") ran")
    return a * b

@st.cache(suppress_st_warning=True)
def expensive_computation(a, b):
    st.write("Cache miss: expensive_computation()", a, ",", b, ") ran")
    time.sleep(2) # This makes the function take 2s to run
    return inner_func(a, b) + 1

a = 2
b = 210
res = expensive_computation(a, b)

st.write("Result:", res)
```

What you see is the usual:

1. The first run results in a cache miss.

2. Every subsequent rerun results in a cache hit.

But now let's try modifying the `inner_func()`:

```
import streamlit as st
import time

def inner_func(a, b):
    st.write("inner_func()", a, ",", b, ") ran")
    return a ** b # 🤝 Changed the * to ** here

@st.cache(suppress_st_warning=True)
def expensive_computation(a, b):
    st.write("Cache miss: expensive_computation()", a, ",", b, ") ran")
    time.sleep(2) # This makes the function take 2s to run
    return inner_func(a, b) + 1

a = 2
b = 21
res = expensive_computation(a, b)

st.write("Result:", res)
```

Even though `inner_func()` is not annotated with `@st.cache`, when we edit its body we cause a "Cache miss" in the outer `expensive_computation()`.

That's because Streamlit always traverses your code and its dependencies to verify that the cached values are still valid. This means that while developing your app you can edit your code freely without worrying about the cache. Any change you make to your app, Streamlit should do the right thing!

Streamlit is also smart enough to only traverse dependencies that belong to your app, and skip over any dependency that comes from an installed Python library.

## Example 5: Use caching to speed up your app across users

Going back to our original function, let's add a widget to control the value of `b`:

```
import streamlit as st
import time

@st.cache(suppress_st_warning=True)
def expensive_computation(a, b):
    st.write("Cache miss: expensive_computation(", a, ", ", b, ") ran")
    time.sleep(2) # This makes the function take 2s to run
    return a * b

a = 2
b = st.slider("Pick a number", 0, 10) # 🤝 Changed this
res = expensive_computation(a, b)

st.write("Result:", res)
```

What you'll see:

- If you move the slider to a number Streamlit hasn't seen before, you'll have a cache miss again. And every subsequent rerun with the same number will be a cache hit, of course.
- If you move the slider back to a number Streamlit has seen before, the cache is hit and the app is fast as expected.

In computer science terms, what is happening here is that @st.cache is memoizing expensive\_computation(a, b).

But now let's go one step further! Try the following:

1. Move the slider to a number you haven't tried before, such as 9.
2. Pretend you're another user by opening another browser tab pointing to your Streamlit app (usually at <http://localhost:8501>)
3. In the new tab, move the slider to 9.

Notice how this is actually a cache hit! That is, you don't actually see the "Cache miss" text on the second tab even though that second user never moved the slider to 9 at any point prior to this.

This happens because the Streamlit cache is global to all users. So everyone contributes to everyone else's performance.

## Example 6: Mutating cached values

As mentioned in the [overview](#) section, the Streamlit cache stores items by reference. This allows the Streamlit cache to support structures that aren't memory-managed by Python, such as TensorFlow objects. However, it can also lead to unexpected behavior — which is why Streamlit has a few checks to guide developers in the right direction. Let's look into those checks now.

Let's write an app that has a cached function which returns a mutable object, and then let's follow up by mutating that object:

```
import streamlit as st
import time

@st.cache(suppress_st_warning=True)
def expensive_computation(a, b):
    st.write("Cache miss: expensive_computation(", a, ", ", b, ") ran")
    time.sleep(2) # This makes the function take 2s to run
    return {"output": a * b} # 🤞 Mutable object

a = 2
b = 21
res = expensive_computation(a, b)

st.write("Result:", res)

res["output"] = "result was manually mutated" # 🤞 Mutated cached value

st.write("Mutated result:", res)
```

When you run this app for the first time, you should see three messages on the screen:

- Cache miss (...)
- Result: {output: 42}
- Mutated result: {output: "result was manually mutated"}

No surprises here. But now notice what happens when you rerun your app (i.e. press **R**):

- Result: {output: "result was manually mutated"}
- Mutated result: {output: "result was manually mutated"}

- Cached object mutated. (...)

So what's up?

What's going on here is that Streamlit caches the output `res` by reference. When you mutated `res["output"]` outside the cached function you ended up inadvertently modifying the cache. This means every subsequent call to `expensive_computation(2, 21)` will return the wrong value!

Since this behavior is usually not what you'd expect, Streamlit tries to be helpful and show you a warning, along with some ideas about how to fix your code.

In this specific case, the fix is just to not mutate `res["output"]` outside the cached function. There was no good reason for us to do that anyway! Another solution would be to clone the result value with `res = deepcopy(expensive_computation(2, 21))`. Check out the section entitled [Fixing caching issues](#) for more information on these approaches and more.

## Advanced caching

In [caching](#), you learned about the Streamlit cache, which is accessed with the `@st.cache` decorator. In this article you'll see how Streamlit's caching functionality is implemented, so that you can use it to improve the performance of your Streamlit apps.

The cache is a key-value store, where the key is a hash of:

1. The input parameters that you called the function with
2. The value of any external variable used in the function
3. The body of the function
4. The body of any function used inside the cached function

And the value is a tuple of:

- The cached output
- A hash of the cached output (you'll see why soon)

For both the key and the output hash, Streamlit uses a specialized hash function that knows how to traverse code, hash special objects, and can have its [behavior customized by the user](#).

For example, when the function `expensive_computation(a, b)`, decorated with `@st.cache`, is executed with `a=2` and `b=21`, Streamlit does the following:

1. Computes the cache key

2. If the key is found in the cache, then:

- Extracts the previously-cached (output, output\_hash) tuple.
- Performs an **Output Mutation Check**, where a fresh hash of the output is computed and compared to the stored `output_hash`.
  - If the two hashes are different, shows a **Cached Object Mutated** warning. (Note: Setting `allow_output_mutation=True` disables this step).

3. If the input key is not found in the cache, then:

- Executes the cached function (i.e. `output = expensive_computation(2, 21)`).
- Calculates the `output_hash` from the function's `output`.
- Stores `key → (output, output_hash)` in the cache.

4. Returns the output.

If an error is encountered an exception is raised. If the error occurs while hashing either the key or the output an `UnhashableTypeError` error is thrown. If you run into any issues, see [fixing caching issues](#).

## The `hash_funcs` parameter

As described above, Streamlit's caching functionality relies on hashing to calculate the key for cached objects, and to detect unexpected mutations in the cached result.

For added expressive power, Streamlit lets you override this hashing process using the `hash_funcs` argument. Suppose you define a type called `FileReference` which points to a file in the filesystem:

```
class FileReference:  
    def __init__(self, filename):  
        self.filename = filename  
  
@st.cache  
def func(file_reference):  
    ...
```

By default, Streamlit hashes custom classes like `FileReference` by recursively navigating their structure. In this case, its hash is the hash of the `filename` property. As long as the file name doesn't change, the hash will remain constant.

However, what if you wanted to have the hasher check for changes to the file's modification time, not just its name? This is possible with `@st.cache`'s `hash_funcs` parameter:

```
class FileReference:  
    def __init__(self, filename):  
        self.filename = filename  
  
def hash_file_reference(file_reference):  
    filename = file_reference.filename  
    return (filename, os.path.getmtime(filename))  
  
@st.cache(hash_funcs={FileReference: hash_file_reference})  
def func(file_reference):  
    ...
```

Additionally, you can hash `FileReference` objects by the file's contents:

```
class FileReference:  
    def __init__(self, filename):  
        self.filename = filename  
  
def hash_file_reference(file_reference):  
    with open(file_reference.filename) as f:  
        return f.read()  
  
@st.cache(hash_funcs={FileReference: hash_file_reference})  
def func(file_reference):  
    ...
```

## ★ Note

Because Streamlit's hash function works recursively, you don't have to hash the contents inside `hash_file_reference`. Instead, you can return a primitive type, in this case the contents of the file, and Streamlit's internal hasher will compute the actual hash from it.

# Typical hash functions

While it's possible to write custom hash functions, let's take a look at some of the tools that Python provides out of the box. Here's a list of some hash functions and when it makes sense to use them.

## Python's `id` function | [Example](#)

- Speed: Fast
- Use case: If you're hashing a singleton object, like an open database connection or a TensorFlow session. These are objects that will only be instantiated once, no matter how many times your script reruns.

## `lambda _: None` | [Example](#)

- Speed: Fast
- Use case: If you want to turn off hashing of this type. This is useful if you know the object is not going to change.

## Python's `hash()` function | [Example](#)

- Speed: Can be slow based the size of the object being cached
- Use case: If Python already knows how to hash this type correctly.

## Custom hash function | [Example](#)

- Speed: N/a
- Use case: If you'd like to override how Streamlit hashes a particular type.

## Example 1: Pass a database connection around

Suppose we want to open a database connection that can be reused across multiple runs of a Streamlit app. For this you can make use of the fact that cached objects are stored by reference to automatically initialize and reuse the connection:

```
@st.cache(allow_output_mutation=True)
def get_database_connection():
    return db.get_connection()
```

With just 3 lines of code, the database connection is created once and stored in the cache. Then, every subsequent time `get_database_connection` is called, the already-created connection object is reused automatically. In other words, it becomes a singleton.

## ★ Tip

Use the `allow_output_mutation=True` flag to suppress the immutability check. This prevents Streamlit from trying to hash the output connection, and also turns off Streamlit's mutation warning in the process.

What if you want to write a function that receives a database connection as input? For that, you'll use `hash_funcs`:

```
@st.cache(hash_funcs={DBConnection: id})
def get_users(connection):
    # Note: We assume that connection is of type DBConnection.
    return connection.execute_sql('SELECT * from Users')
```

Here, we use Python's built-in `id` function, because the connection object is coming from the Streamlit cache via the `get_database_connection` function. This means that the same connection instance is passed around every time, and therefore it always has the same id. However, if you happened to have a second connection object around that pointed to an entirely different database, it would still be safe to pass it to `get_users` because its id is guaranteed to be different than the first id.

These design patterns apply any time you have an object that points to an external resource, such as a database connection or Tensorflow session.

## Example 2: Turn off hashing for a specific type

You can turn off hashing entirely for a particular type by giving it a custom hash function that returns a constant. One reason that you might do this is to avoid hashing large, slow-to-hash objects that you know are not going to change. For example:

```
@st.cache(hash_funcs={pd.DataFrame: lambda _: None})
def func(huge_constant_dataframe):
    ...
```

When Streamlit encounters an object of this type, it always converts the object into `None`, no matter which instance of `FooType` its looking at. This means all instances are hash to the same value, which effectively cancels out the hashing mechanism.

## Example 3: Use Python's hash() function

Sometimes, you might want to use Python's default hashing instead of Streamlit's. For example, maybe you've encountered a type that Streamlit is unable to hash, but it's hashable with Python's built-in `hash()` function:

```
@st.cache(hash_funcs={FooType: hash})  
def func(...):  
    ...
```

Was this page helpful?

 Yes

 No

 [Suggest edits](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

[← Previous: Configuration](#)

[Next: Experimental cache primitives →](#)

[Home](#)

[Contact Us](#)

[Community](#)



Copyright © 2022, Streamlit Inc.

# Documentation



Search



Streamlit library

- Get started



- API reference



- Advanced features



- Components



- Changelog

- Cheat sheet



Streamlit Cloud

- Get started



- Trust and Security

- Release notes

- Troubleshooting



Knowledge base

- Tutorials



- Using Streamlit

- Streamlit Components

- Installing dependencies
- Deployment issues

[Home](#) / [Streamlit library](#) / [Changelog](#)

# Changelog

This page lists highlights, bug fixes, and known issues for official Streamlit releases. If you're looking for information about nightly releases, beta features, or experimental features, see [Try pre-release features](#).

## ★ Tip

To upgrade to the latest version of Streamlit, run:

```
pip install --upgrade streamlit
```

## Version 1.9.0

*Release date: May 4, 2022*

### Notable Changes

- 🎨 `st.json` now supports a keyword-only argument, `expanded` on whether the JSON should be expanded by default (defaults to `True`).
- 🏃 More performance improvements from reducing redundant work each script run.

### Other Changes

- 🐛 Widgets when `disabled` is set/unset will maintain its value (#4527).
- 🖌 Experimental feature to increase the speed of reruns using configuration `runner.fastReruns`. See #4628 for the known issues in enabling this feature.
- 📈 DataFrame timestamps support UTC offset (in addition to time zone notation) (#4669).

# Version 1.8.0

Release date: March 24, 2022

## Notable Changes

- 🏃 Dataframes should see performance improvements (#4463).

## Other Changes

- ⏳ `st.slider` handles timezones better by removing timezone conversions on the backend (#4348).
- 🧑 Design improvements to our header (#4496).

# Version 1.7.0

Release date: March 3, 2022

## Highlights

- Introducing `st.snow`, celebrating our acquisition by Snowflake! See more information in our blog post.

# Version 1.6.0

Release date: Feb 24, 2022

## Other Changes

- ⚡ WebSocket compression is now disabled by default, which will improve CPU and latency performance for large dataframes. You can use the `server.enablewebsocketCompression` configuration option to re-enable it if you find the increased network traffic more impactful.
- ✅ Radio and checkboxes improve focus on Keyboard navigation (#4308).

# Version 1.5.0

Release date: Jan 27, 2022

## Notable Changes

- 🌟 Favicon defaults to a PNG to allow for transparency (#4272).

- ⚠️ Select Slider Widget now has the `disabled` parameter that removes interactivity (completing all of our widgets) (#4314).

## Other Changes

- 📄 Improvements to our markdown library to provide better support for HTML (specifically nested HTML) (#4221).
- 📄 Expanders maintain their expanded state better when multiple expanders are present (#4290).
- 📁 Improved file uploader and camera input to call its `on_change` handler only when necessary (#4270).

# Version 1.4.0

Release date: Jan 13, 2022

## Highlights

- 📸 Introducing `st.camera_input` for uploading images straight from your camera.

## Notable Changes

- ⚠️ Widgets now have the `disabled` parameter that removes interactivity.
- 🗑️ Clear `st.experimental_memo` and `st.experimental_singleton` programmatically by using the `clear()` method on a cached function.
- 📲 Developers can now configure the maximum size of a message to accommodate larger messages within the Streamlit application. See `server.maxMessageSize`.
- 🎉 We formally added support for Python 3.10.

## Other Changes

- 🤔 Calling `str` or `repr` on `threading.current_thread()` does not cause a RecursionError (#4172).
- 📹 Gracefully stop screencast recording when user removes permission to record (#4180).
- 🌈 Better scale images by using a higher-quality image bilinear resampling algorithm (#4159).

# Version 1.3.0

## Notable Changes

-  Support for NumPy values in `st.metric`.
-  Support for Mesh Layers in PyDeck.
-  Updated Plotly chart version to support the latest features.
-  `st.spinner` element has visual animated spinner.
-  `st.caption` supports HTML in text with `unsafe_allow_html` parameter.

## Other Changes

-  Bug fix: Allow `st.session_state` to be used to set `number_input` values with no warning (#4047).
-  Bug fix: Fix footer alignment in wide mode (#4035).
-  Bug fix: Better support for Graphviz and Bokeh charts in containers (columns, expanders, etc.) (#4039).
-  Bug fix: Support inline data values in Vega-Lite (#4070).
-  Types: Updated type annotations for experimental memo and singleton decorators.
-  Types: Improved type annotations for `st.selectbox`, `st.select_slider`, `st.radio`, `st.number_input`, and `st.multiselect`.

# Version 1.2.0

Release date: Nov 11, 2021

## Notable Changes

-  `st.text_input` and `st.text_area` now have a `placeholder` parameter to display text when the field is empty.
-  Viewers can now resize the input box in `st.text_area`.
-  Streamlit can auto-reload when files in sub-directories change.
-  We've upgraded Bokeh support to 2.4.1! We recommend updating your Bokeh library to 2.4.1 to maintain functionality. Going forward, we'll let you know if there's a mismatch in your Bokeh version via an error prompt.

- Developers can access secrets via attribute notation (e.g. `st.secrets.key`) vs `st.secrets["key"]` just like session state.
- Publish type annotations according to PEP 561. Users now get type annotations for Streamlit when running `mypy` (#4025).

## Other Changes

- Visual fixes (#3863, #3995, #3926, #3975).
- Fixes to the hamburger menu (#3968).
- Ability to print session state (#3970).

# Version 1.1.0

Release date: Oct 21, 2021

## Highlights

- Memory improvements: Streamlit apps allocate way less memory over time now.

## Notable Changes

- Apps automatically rerun now when the content of `secrets.toml` changes (before this you had to refresh the page manually).

## Other Changes

- Redirected some links to our brand-new docs site, e.g. in exceptions.
- Bug fix: Allow initialization of range slider with session state (#3586).
- Bug fix: Refresh chart when using `add_rows` with `datetime` index (#3653).
- Added some more type annotation in our codebase (#3908).

# Version 1.0.0

Release date: Oct 5, 2021

## Highlights

- Announcing Streamlit 1.0! To read more about check out our 1.0 blog post.

## Other Changes

- 🐛 Fixed an issue where using `df.dtypes` to show datatypes for a DF fails while using Arrow (#3709),  
Image captions stay within image width and are readable (#3530).

# Version 0.89.0

Release date: Sep 22, 2021

## Highlights

- 💰 Introducing `st.experimental_memo` and `experimental_singleton`, a new primitive for caching! See our blog post.
- 🍔 Streamlit allows developers to configure their hamburger menu to be more user-centric.

## Notable Changes

- 💅 We updated our UI to a more polished look with a new font.
- 🎨 We now support `theme.base` in the theme object when it's sent to custom components.
- 🧠 We've modified session state to reset widgets if any of their arguments changed even if they provide a key.
  - Some widget behavior may have changed, but we believe this change makes the most sense. We have added a section to our documentation describing how they behave.

## Other Changes

- 🐛 Bug fixes: Support svgs from a URL (#3809) and that do not start with `<svg>` tag (#3789).

# Version 0.88.0

Release date: Sep 2, 2021

## Highlights

- ⬇️ Introducing `st.download_button`, a new button widget for easily downloading files.

## Notable Changes

- 📝 We made changes to improve the redacted exception experience on Streamlit Cloud. When `client.showErrorDetails=true` exceptions display the Error Type and the Traceback, but redact the actual error text to prevent data leaks.

## Version 0.87.0

Release date: Aug 19, 2021

### Highlights

- 🚧 Introducing `st.metric`, an API for displaying KPIs. Check out the demo app showcasing the functionality.

### Other Changes

- 🐞 **Bug Fixes:** File uploader retains state upon expander closing (#3557), setIn Error with `st.empty` (#3659), Missing IFrame embeds in docs (#3706), Fix error writing certain PNG files (#3597).

## Version 0.86.0

Release date: Aug 5, 2021

### Highlights

- 🎓 Our layout primitives are graduating from beta! You can now use `st.columns`, `st.container` and `st.expander` without the `beta_` prefix.

### Notable Changes

- 🖥️ When using `st.columns`, columns will stack vertically when viewport size <640px so that column layout on smaller viewports is consistent and cleaner. (#3594).

### Other Changes

- 🐞 **Bug fixes:** Fixed `st.date_input` crashes if its empty (#3194), Opening files with utf-8(#3022), `st.select_slider` resets its state upon interaction (#3600).

## Version 0.85.0

Release date: Jul 22, 2021

## Highlights

- 🌟 Streamlit now uses Apache Arrow for serializing data frames when they are sent from Streamlit server to the front end. See our blog post.
  - (Users who wish to continue using the legacy data frame serialization can do so by setting the `dataFrameSerialization` config option to `"legacy"` in their `config.toml` ).

## Other Changes

- 🐛 Bug fixes: Unresponsive pydeck example (#3395), JSON parse error message (#2324), Tooltips rendering (#3300), Colorpicker not working on Streamlit Sharing (#2689).

# Version 0.84.0

Release date: Jul 1, 2021

## Highlights

- 🧠 Introducing `st.session_state` and widget callbacks to allow you to add statefulness to your apps. Check out the blog post

## Notable Changes

- ✨ `st.text_input` now has an `autocomplete` parameter to allow password managers to be used

## Other Changes

- Using `st.set_page_config` to assign the page title no longer appends "Streamlit" to that title (#3467)
- NumberInput: disable plus/minus buttons when the widget is already at its max (or min) value (#3493)

# Version 0.83.0

Release date: Jun 17, 2021

## Highlights

- 🚦 Updates to Streamlit docs to include step-by-step guides which demonstrate how to connect Streamlit apps to various databases & APIs

## Notable Changes

-  `st.form` now has a `clear_on_submit` parameter which "resets" all the form's widgets when the form is submitted.

## Other Changes

- Fixed bugs regarding file encodings (#3320, #3108, #2731)

# Version 0.82.0

Release date: May 13, 2021

## Notable Changes

-  Improvements to memory management by forcing garbage collection between script runs.

# Version 0.81.1

Release date: Apr 29, 2021

## Highlights

-  Introducing `st.form` and `st.form_submit_button` to allow you to batch input widgets. Check out our blog post
-  Introducing `st.caption` so you can add explainer text anywhere in your apps.
-  Updates to Theming, including ability to build a theme that inherits from any of our default themes.
-  Improvements to deployment experience to Streamlit sharing from the app menu.

## Other changes

- Support for binary files in Custom Components (#3144)

# Version 0.80.0

Release date: Apr 8, 2021

## Highlights

-  Streamlit now supports Secrets management for apps deployed to Streamlit Sharing!

- ⚓ Titles and headers now come with automatically generated anchor links. Just hover over any title and click the 🔗 to get the link!

## Other changes

- Added `allow-downloads` capability to custom components (#3040)
- Fixed markdown tables in dark theme (#3020)
- Improved color picker widget in the Custom Theme dialog (#2970)

# Version 0.79.0

Release date: Mar 18, 2021

## Highlights

- 🌈 Introducing support for custom themes. Check out our blog post
- 🌐 This release also introduces dark mode!
- 💡 Support for tooltips on all input widgets

## Other changes

- Fixed bugs regarding file encodings (#1936, #2606) and caching functions (#2728)

# Version 0.78.0

Release date: Mar 4, 2021

## Features

- If you're in the Streamlit for Teams beta, we made a few updates to how secrets work. Check the beta docs for more info!
- Dataframes now displays timezones for all DateTime and Time columns, and shows the time with the timezone applied, rather than in UTC

## Notable Bug Fixes

- Various improvement to column alignment in `st.beta_columns`
- Removed the long-deprecated `format` param from `st.image`, and replaced with `output_format`.

# Version 0.77.0

Release date: Feb 23, 2021

## Features

- Added a new config option `client.showErrorDetails` allowing the developer to control the granularity of error messages. This is useful for when you deploy an app, and want to conceal from your users potentially-sensitive information contained in tracebacks.

## Notable bug fixes

- Fixed bug where `st.image` wasn't rendering certain kinds of SVGs correctly.
- Fixed regression where the current value of an `st.slider` was only shown on hover.

# Version 0.76.0

Release date: February 4, 2021

## Notable Changes

- 🎨 `st.color_picker` is now out of beta. This means the old `beta_color_picker` function, which was marked as deprecated for the past 3 months, has now been replaced with `color_picker`.
- 🚨 Display a warning when a Streamlit script is run directly as `python script.py`.
- `st.image`'s `use_column_width` now defaults to an `auto` option which will resize the image to the column width if the image exceeds the column width.
- ✘ Fixed bugs (2437 and 2247) with content getting cut off within a `st.beta_expander`
- 📦 Fixed a bug in `st.dataframe` where the scrollbar overlapped with the contents in the last column.
- 🗂️ Fixed a bug for `st.file_uploader` where file data returned was not the most recently uploaded file.
- ✅ Fixed bugs (2086 and 2556) where some LaTeX commands were not rendering correctly.

# Version 0.75.0

Release date: January 21, 2021

## Notable Changes

-  `st.empty` previously would clear the component at the end of the script. It has now been updated to clear the component instantly.
-  Previously in wide mode, we had thin margins around the webpage. This has now been increased to provide a better visual experience.

## Version 0.74.0

Release date: January 6, 2021

### Notable Changes

-  `st.file_uploader`. has been stabilized and the deprecation warning and associated configuration option (`deprecation.showfileUploaderEncoding`) has been removed.
-  `st.bokeh_chart` is no longer duplicated when the page loads.
-  Fixed page icon to support emojis with variants (i.e. 🌟 vs 🌟) or dashes (i.e. ☽ - crescent-moon).

## Version 0.73.0

Release date: December 17, 2020

### Notable Changes

-  Streamlit can now be installed on Python 3.9. Streamlit components are not yet compatible with Python 3.9 and must use version 3.8 or earlier.
-  Streamlit Components now allows same origin, enabling features provided by the browser such as a webcam component.
-  Fix Streamlit sharing deploy experience for users running on Git versions 2.7.0 or earlier.
-  Handle unexpected closing of uploaded files for `st.file_uploader`.

## Version 0.72.0

Release date: December 2, 2020

### Notable Changes

-  Establish a framework for theming and migrate existing components.

- Improve the sidebar experience for mobile devices.
- Update `st.file_uploader` to reduce reruns.

## Version 0.71.0

Release date: November 11, 2020

### Notable Changes

- Updated `st.file_uploader` to automatically reset buffer on app reruns.
- Optimize the default rendering of charts and reduce issues with the initial render.

## Version 0.70.0

Release date: October 28, 2020

### Notable Changes

- `st.set_page_config` and `st.color_picker` have now been moved into the Streamlit namespace. These will be removed from beta January 28th, 2021. Learn more about our beta process [here](#).
- Improve display of bar charts for discrete values.

## Version 0.69.0

Release date: October 15, 2020

### Highlights:

- Introducing Streamlit sharing, the best way to deploy, manage, and share your public Streamlit apps—for free. Read more about it on our blog post or sign up [here](#)!
- Added `st.experimental_rerun` to programmatically re-run your app. Thanks SimonBiggs!

### Notable Changes

- Better support across browsers for start and stop times for `st.video`.
- Bug fix for intermittently failing media files

- 📦 Bug fix for custom components compatibility with Safari. Make sure to upgrade to the latest streamlit-component-lib.

## Version 0.68.0

Release date: October 8, 2020

### Highlights:

- # Introducing new layout options for Streamlit! Move aside, vertical layout. Make a little space for... horizontal layout! Check out our blog post.
- 📁 File uploader redesigned with new functionality for multiple files uploads and better support for working with uploaded files. This may cause breaking changes. Please see the new api in our documentation

### Notable Changes

- 🎈 `st.balloon` has gotten a facelift with nicer balloons and smoother animations.
- ⚡ Breaking Change: Following the deprecation of `st.deck_gl_chart` in January 2020, we have now removed the API completely. Please use `st.pydeck_chart` instead.
- ⚡ Breaking Change: Following the deprecation of `width` and `height` for `st.altair_chart`, `st.graphviz_chart`, `st.plotly_chart`, and `st.vega_lite_chart` in January 2020, we have now removed the args completely. Please set the width and height in the respective charting library.

## Version 0.67.0

Release date: September 16, 2020

### Highlights:

- 🦢 Streamlit Components can now return bytes to your Streamlit App. To create a component that returns bytes, make sure to upgrade to the latest streamlit-component-lib.

### Notable Changes

- ✋ Deprecation warning: Beginning December 1st, 2020 `st.pyplot()` will require a figure to be provided. To disable the deprecation warning, please set `deprecation.showPyplotGlobalUse` to `False`
- 🛡️ `st.multiselect` and `st.select` are now lightning fast when working with large datasets.  
Thanks masa3141!

# Version 0.66.0

Release date: September 1, 2020

## Highlights:

- `st.write` is now available for use in the sidebar!
- A slider for distinct or non-numerical values is now available with `st.select_slider`.
- # Streamlit Components can now return dataframes to your Streamlit App. Check out our SelectableDataTable example.
- The Streamlit Components library used in our Streamlit Component template is now available as an npm package (`streamlit-component-lib`) to simplify future upgrades to the latest version. Existing components do not need to migrate.

## Notable Changes

- Support `StringDtype` from pandas version 1.0.0
- Support for running Streamlit on Unix sockets

# Version 0.65.0

Release date: August 12, 2020

## Highlights:

- Ability to set page title, favicon, sidebar state, and wide mode via `st.beta_set_page_config()`. See our documentation for details.
- Add stateful behaviors through the use of query parameters with `st.experimental_set_query_params` and `st.experimental_get_query_params`. Thanks @zhaooyue!
- Improved pandas dataframe support for `st.radio`, `st.selectbox`, and `st.multiselect`.
- Break out of your Streamlit app with `st.stop`.
- Inline SVG support for `st.image`.

## Callouts:

- Deprecation Warning: The `st.image` parameter `format` has been renamed to `output_format`.

# Version 0.64.0

Release date: July 23, 2020

## Highlights:

- 📈 Default matplotlib to display charts with a tight layout. To disable this, set `bbox_inches` to `None`, inches as a string, or a `Bbox`
- 🗃 Deprecation warning for automatic encoding on `st.file_uploader`
- 🚧 If `gatherUserStats` is `False`, do not even load the Segment library. Thanks @tanmaylaud!

# Version 0.63.0

Release date: July 13, 2020

## Highlights:

- 🎨 Support for Streamlit Components!!! See documentation for more info.
- ⏰ Support for datetimes in `st.slider`. And, of course, just like any other value you use in `st.slider`, you can also pass in two-element lists to get a datetime range slider.

# Version 0.62.0

Release date: June 21, 2020

## Highlights:

- 💡 Ability to turn websocket compression on/off via the config option `server.enableWebSocketCompression`. This is useful if your server strips HTTP headers and you do not have access to change that behavior.
- 🔑 Out-of-the-box support for CSRF protection using the Cookie-to-header token technique. This means that if you're serving your Streamlit app from multiple replicas you'll need to configure them to use the same cookie secret with the `server.cookieSecret` config option. To turn XSRF protection off, set `server.enableXsrfProtection=false`.

## Notable bug fixes:

- 🖼 Added a grace period to the image cache expiration logic in order to fix multiple related bugs where images sent with `st.image` or `st.pyplot` were sometimes missing.

# Version 0.61.0

Release date: June 2, 2020

## Highlights:

- 📅 Support for date ranges in `st.date_picker`. See docs for more info, but the TLDR is: just pass a list/tuple as the default date and it will be interpreted as a range.
- 🗣 You can now choose whether `st.echo` prints the code above or below the output of the echoed block. To learn more, refer to the `code_location` argument in the docs.
- 📦 Improved `@st.cache` support for Keras models and Tensorflow `saved_models`.

# Version 0.60.0

Release date: May 18, 2020

## Highlights:

- ↑ Ability to set the height of an `st.text_area` with the `height` argument (expressed in pixels). See docs for more.
- 🔢 Ability to set the maximum number of characters allowed in `st.text_area` or `st.text_input`. Check out the `max_chars` argument in the docs.
- gMaps Better DeckGL support for the H3 geospatial indexing system. So now you can use things like `H3HexagonLayer` in `st.pydeck_chart`.
- 📦 Improved `@st.cache` support for PyTorch TensorBase and Model.

# Version 0.59.0

Release date: May 05, 2020

## Highlights:

- 🎨 New color-picker widget! Use it with `st.beta_color_picker()`
- ✍️ Introducing `st.beta_*` and `st.experimental_*` function prefixes, for faster Streamlit feature releases. See docs for more info.
- 📦 Improved `@st.cache` support for SQL Alchemy objects, CompiledFFI, PyTorch Tensors, and `builtins.mappingproxy`.

# Version 0.58.0

Release date: April 22, 2020

## Highlights:

- Made `st.selectbox` filtering case-insensitive.
- Better support for Tensorflow sessions in `@st.cache`.
- Changed behavior of `st.pyplot` to auto-clear the figure only when using the global Matplotlib figure (i.e. only when calling `st.pyplot()` rather than `st.pyplot(fig)`).

# Version 0.57.0

Release date: March 26, 2020

## Highlights:

- Ability to set expiration options for `@st.cache`'ed functions by setting the `max_entries` and `ttl` arguments. See docs.
- Improved the machinery behind `st.file_uploader`, so it's much more performant now! Also increased the default upload limit to 200MB (configurable via `server.max_upload_size`).
- The `server.address` config option now *binds* the server to that address for added security.
- Even more details added to error messages for `@st.cache` for easier debugging.

# Version 0.56.0

Release date: February 15, 2020

## Highlights:

- Improved error messages for `st.cache`. The errors now also point to the new caching docs we just released. Read more here!

## Breaking changes:

- As announced last month, **Streamlit no longer supports Python 2**. To use Streamlit you'll need Python 3.5 or above.

# Version 0.55.0

Release date: February 4, 2020

## Highlights:

- Ability to record screencasts directly from Streamlit! This allows you to easily record and share explanations about your models, analyses, data, etc. Just click `≡` then "Record a screencast". Give it a try!

# Version 0.54.0

Release date: January 29, 2020

## Highlights:

- Support for password fields! Just pass `type="password"` to `st.text_input()`.

## Notable fixes:

- Numerous `st.cache` improvements, including better support for complex objects.
- Fixed cross-talk in sidebar between multiple users.

## Breaking changes:

- If you're using the SessionState `hack` Gist, you should re-download it! Depending on which hack you're using, here are some links to save you some time:
  - `SessionState.py`
  - `st_state_patch.py`

# Version 0.53.0

Release date: January 14, 2020

## Highlights:

- Support for all DeckGL features! Just use Pydeck instead of `st.deck_gl_chart`. To do that, simply pass a PyDeck object to `st.pydeck_chart`, `st.write`, or magic.

*Note that as a **preview release** things may change in the near future. Looking forward to hearing input from the community before we stabilize the API!*

The goals is for this to replace `st.deck_gl_chart`, since it is does everything the old API did and much more!

-  Better handling of Streamlit upgrades while developing. We now auto-reload the browser tab if the app it is displaying uses a newer version of Streamlit than the one the tab is running.
-  New favicon, with our new logo!

## Notable fixes:

- Magic now works correctly in Python 3.8. It no longer causes docstrings to render in your app.

## Breaking changes:

- Updated how we calculate the default width and height of all chart types. We now leave chart sizing up to your charting library itself, so please refer to the library's documentation.

As a result, the `width` and `height` arguments have been deprecated from most chart commands, and `use_container_width` has been introduced everywhere to allow you to make charts fill as much horizontal space as possible (this used to be the default).

# Version 0.52.0

Release date: December 20, 2019

## Highlights:

-  Preview release of the file uploader widget. To try it out just call `st.file_uploader`!

Note that as a **preview release** things may change in the near future. Looking forward to hearing input from the community before we stabilize the API!

-  Support for emoji codes in `st.write` and `st.markdown`! Try it out with `st.write("Hello :wave:")`.

## Breaking changes:

-  `st.pyplot` now clears figures by default, since that's what you want 99% of the time. This allows you to create two or more Matplotlib charts without having to call `pyplot.clf` every time. If you want to turn this behavior off, use `st.pyplot(clear_figure=False)`
-  `st.cache` no longer checks for input mutations. This is the first change of our ongoing effort to simplify the caching system and prepare Streamlit for the launch of other caching primitives like Session State!

# Version 0.51.0

Release date: November 30, 2019

## Highlights:

- 🐶 You can now tweak the behavior of the file watcher with the config option `server.fileWatcherType`. Use it to switch between:
  - `auto` (default) : Streamlit will attempt to use the watchdog module, and falls back to polling if watchdog is not available.
  - `watchdog` : Force Streamlit to use the watchdog module.
  - `poll` : Force Streamlit to always use polling.
  - `none` : Streamlit will not watch files.

## Notable bug fixes:

- Fix the "keyPrefix" option in static report sharing #724
- Add support for `getColorX` and `getTargetColorX` to DeckGL Chart #718
- Fixing Tornado on Windows + Python 3.8 #682
- Fall back on webbrowser if `xdg-open` is not installed on Linux #701
- Fixing number input spin buttons for Firefox #683
- Fixing CTRL+ENTER on Windows #699
- Do not automatically create credential file when in headless mode #467

# Version 0.50.1

Release date: November 10, 2019

## Highlights:

- 🐱 SymPy support and ability to draw mathematical expressions using LaTeX! See `st.latex`, `st.markdown`, and `st.write`.
- 💡 You can now set config options using environment variables. For example, `export STREAMLIT_SERVER_PORT=9876`.
- 🐱 Ability to call `streamlit run` directly with Github and Gist URLs. No need to grab the "raw" URL first!

- 📄 Cleaner exception stack traces. We now remove all Streamlit-specific code from stack traces originating from the user's app.

## Version 0.49.0

Release date: October 23, 2019

### Highlights:

- 💯 New input widget for entering numbers with the keyboard: `st.number_input()`
- 🎵 Audio/video improvements: ability to load from a URL, to embed YouTube videos, and to set the start position.
- 🤝 You can now (once again) share static snapshots of your apps to S3! See the S3 section of `streamlit config show` to set it up. Then share from top-right menu.
- 🛡️ Use `server.baseUrlPath` config option to set Streamlit's URL to something like `http://domain.com/customPath`.

### Notable bug fixes:

- Fixes numerous Windows bugs, including Issues #339 and #401.

## Version 0.48.0

Release date: October 12, 2019

### Highlights:

- 🔐 Ability to set config options as command line flags or in a local config file.
- ⚡ You can now maximize charts and images!
- ⚡ Streamlit is now much faster when writing data in quick succession to your app.
- 🌐 Ability to blacklist folder globs from "run on save" and `@st.cache` hashing.
- 🎯 Improved handling of widget state when Python file is modified.
- 🎩 Improved HTML support in `st.write` and `st.markdown`. HTML is still unsafe, though!

### Notable bug fixes:

- Fixes `@st.cache` bug related to having your Python environment on current working directory. Issue #242
- Fixes loading of root url  on Windows. Issue #244

## Version 0.47.0

Release date: October 1, 2019

### Highlights:

- 🎨 New hello.py showing off 4 glorious Streamlit apps. Try it out!
- 🔄 Streamlit now automatically selects an unused port when 8501 is already in use.
- 🎁 Sidebar support is now out of beta! Just start any command with `st.sidebar.` instead of `st.`
- ⚡ Performance improvements: we added a cache to our websocket layer so we no longer re-send data to the browser when it hasn't changed between runs
- 🖥 Our "native" charts `st.line_chart`, `st.area_chart` and `st.bar_chart` now use Altair behind the scenes
- 💪 Improved widgets: custom st.slider labels; default values in multiselect
- 🧑 The filesystem watcher now ignores hidden folders and virtual environments
- 🌈 Plus lots of polish around caching and widget state management

### Breaking change:

- 🛡 We have temporarily disabled support for sharing static "snapshots" of Streamlit apps. Now that we're no longer in a limited-access beta, we need to make sure sharing is well thought through and abides by laws like the DMCA. But we're working on a solution!

## Version 0.46.0

Release date: September 19, 2019

### Highlights:

- ✨ Magic commands! Use `st.write` without typing `st.write`. See <https://docs.streamlit.io/en/latest/api.html#magic-commands>
- 🎯 New `st.multiselect` widget.

- 🌱 Fixed numerous install issues so now you can use `pip install streamlit` even in Conda! We've therefore deactivated our Conda repo.
- 🐞 Multiple bug fixes and additional polish in preparation for our launch!

### Breaking change:

- 🚨 HTML tags are now blacklisted in `st.write` / `st.markdown` by default. More information and a temporary work-around at: <https://github.com/streamlit/streamlit/issues/152>

## Version 0.45.0

Release date: August 28, 2019

### Highlights:

- 😱 Experimental support for `sidebar`! Let us know if you want to be a beta tester.
- 🎁 Completely redesigned `st.cache`! Much more performant, has a cleaner API, support for caching functions called by `@st.cached` functions, user-friendly error messages, and much more!
- 🖼 Lightning fast `st.image`, ability to choose between JPEG and PNG compression, and between RGB and BGR (for OpenCV).
- 💡 Smarter API for `st.slider`, `st.selectbox`, and `st.radio`.
- 🤖 Automatically fixes the Matplotlib backend -- no need to edit `.matplotlibrc`

## Version 0.44.0

Release date: July 28, 2019

### Highlights:

- ⚡ Lightning-fast reconnect when you do a ctrl-c/rerun on your Streamlit code
- 🔔 Useful error messages when the connection fails
- 💎 Fixed multiple bugs and improved polish of our newly-released interactive widgets

## Version 0.43.0

## Highlights:

- ⚡ Support for interactive widgets! 🎉

# Version 0.42.0

Release date: July 1, 2019

## Highlights:

- 💾 Ability to save Vega-Lite and Altair charts to SVG or PNG
- ⚡ We now cache JS files in your browser for faster loading
- 🚫 Improvements to error-handling inside Streamlit apps

# Version 0.41.0

Release date: June 24, 2019

## Highlights:

- ↗️ Greatly improved our support for named datasets in Vega-Lite and Altair
- 🤖 Added ability to ignore certain folders when watching for file changes. See the `server.folderWatchBlacklist` config option.
- 👾 More robust against syntax errors on the user's script and imported modules

# Version 0.40.0

Release date: June 10, 2019

## Highlights:

- Streamlit is more than 10x faster. Just save and watch your analyses update instantly.
- We changed how you run Streamlit apps: `$ streamlit run your_script.py [script args]`
- Unlike the previous versions of Streamlit, `streamlit run [script] [script args]` creates a server (now you don't need to worry if the proxy is up). To kill the server, all you need to do is hit **Ctrl+c**.

## Why is this so much faster?

Now, Streamlit keeps a single Python session running until you kill the server. This means that Streamlit can re-run your code without kicking off a new process; imported libraries are cached to memory. An added bonus is that `st.cache` now caches to memory instead of to disk.

## What happens if I run Streamlit the old way?

If you run `$ python your_script.py` the script will execute from top to bottom, but won't produce a Streamlit app.

## What are the limitations of the new architecture?

- To switch Streamlit apps, first you have to kill the Streamlit server with **Ctrl-c**. Then, you can use `streamlit run` to generate the next app.
- Streamlit only works when used inside Python files, not interactively from the Python REPL.

## What else do I need to know?

- The strings we print to the command line when **liveSave** is on have been cleaned up. You may need to adjust any RegEx that depends on those.
- A number of config options have been renamed:

Old config	New config
proxy.isRemote	server.headless
proxy.liveSave	server.liveSave
proxy.runOnSave	server.runOnSave
proxy.watchFileSystem	server.runOnSave
proxy.enableCORS	server.enableCORS
proxy.port	server.port
browser.proxyAddress	browser.serverAddress
browser.proxyPort	browser.serverPort
client.waitForProxySecs	n/a
client.throttleSecs	n/a

client.tryToOutliveProxy	n/a
client.proxyAddress	n/a
client.proxyPort	n/a
proxy.autoCloseDelaySecs	n/a
proxy.reportExpirationSecs	n/a

## What if something breaks?

If the new Streamlit isn't working, please let us know by Slack or email. You can downgrade at any time with these commands:

```
pip install --upgrade streamlit==0.37
```

```
conda install streamlit=0.37
```

## What's next?

Thank you for staying with us on this journey! This version of Streamlit lays the foundation for interactive widgets, a new feature of Streamlit we're really excited to share with you in the next few months.

## Version 0.36.0

*Release date: May 03, 2019*

### Highlights

- 🚀 `st.progress()` now also accepts floats from 0.0–1.0
- 🎉 Improved rendering of long headers in DataFrames
- 🔒 Shared apps now default to HTTPS

## Version 0.35.0

## Highlights

- 📷 Bokeh support! Check out docs for [st.bokeh\\_chart](#)
- ⚡ Improved the size and load time of saved apps
- ⚾ Implemented better error-catching throughout the codebase

## Was this page helpful?

 Yes       No

 [Suggest edits](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

[← Previous: Components](#)

[Next: Cheat sheet →](#)

[Home](#)[Contact Us](#)[Community](#)

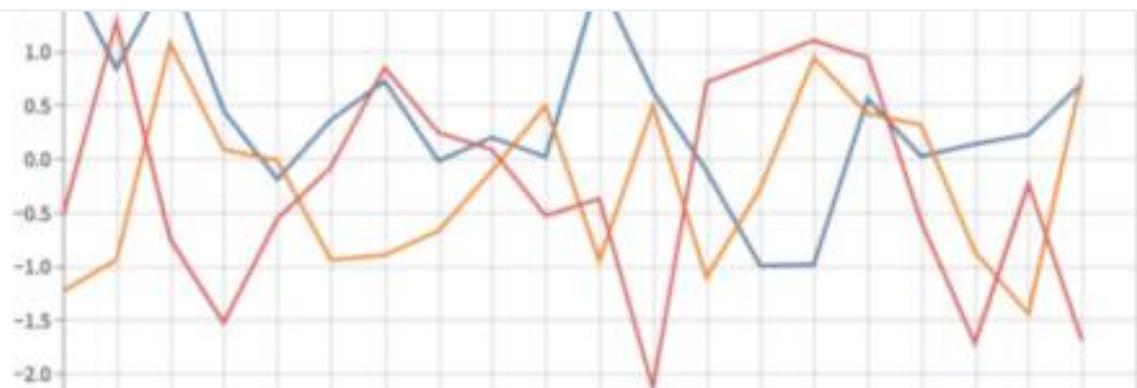


Copyright © 2022, Streamlit Inc.



# Chart elements

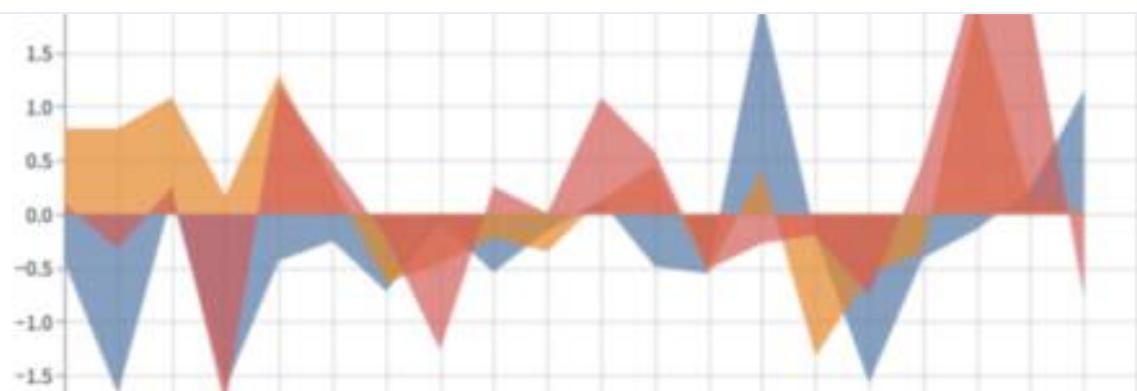
Streamlit supports several different charting libraries, and our goal is to continually add support for more. Right now, the most basic library in our arsenal is [Matplotlib](#). Then there are also interactive charting libraries like [Vega Lite](#) (2D charts) and [deck.gl](#) (maps and 3D charts). And finally we also provide a few chart types that are "native" to Streamlit, like `st.line_chart` and `st.area_chart`.



## Simple line charts

Display a line chart.

```
st.line_chart(my_data_frame)
```



## Simple area charts

Display an area chart.

```
st.area_chart(my_data_frame)
```





## Simple bar charts

Display a bar chart.

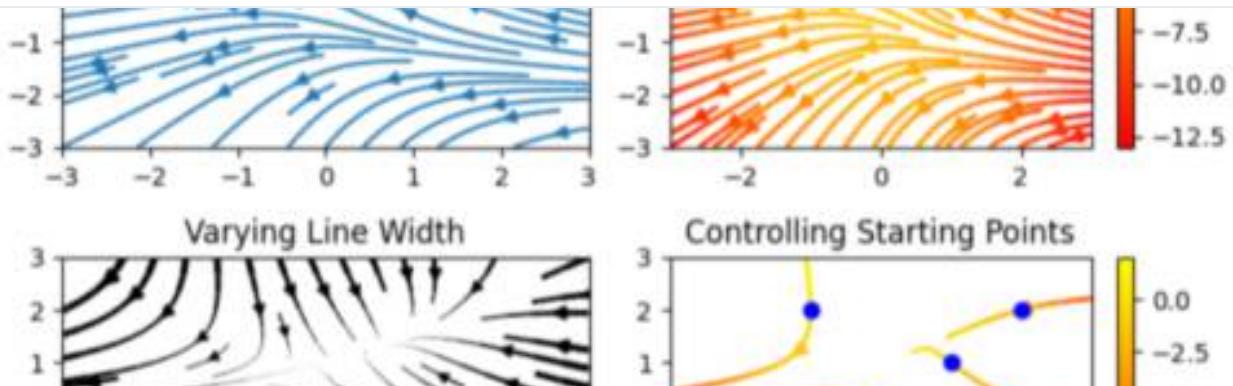
```
st.bar_chart(my_data_frame)
```



## Scatterplots on maps

Display a map with points on it.

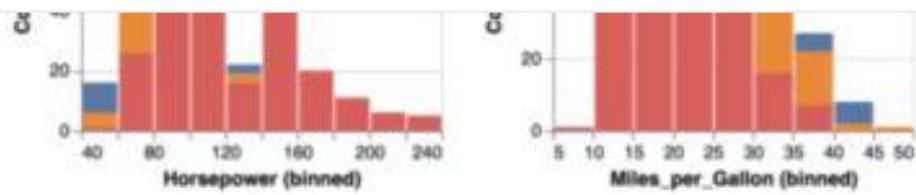
```
st.map(my_data_frame)
```



## Matplotlib

Display a matplotlib.pyplot figure.

```
st.pyplot(my_mpl_figure)
```

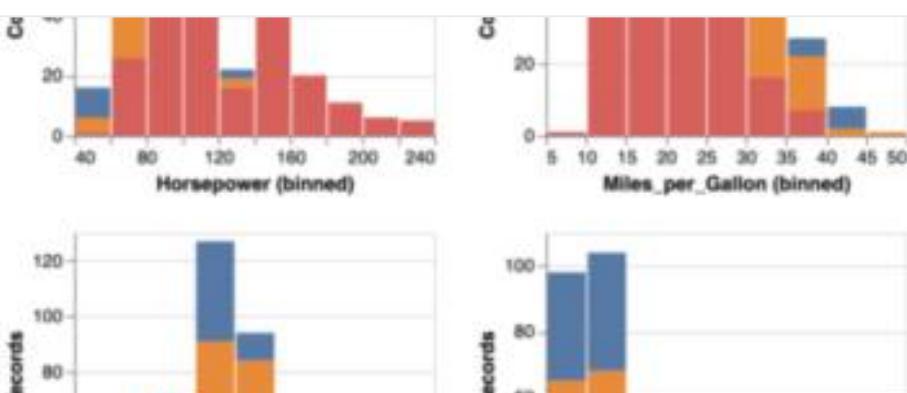




## Altair

Display a chart using the Altair library.

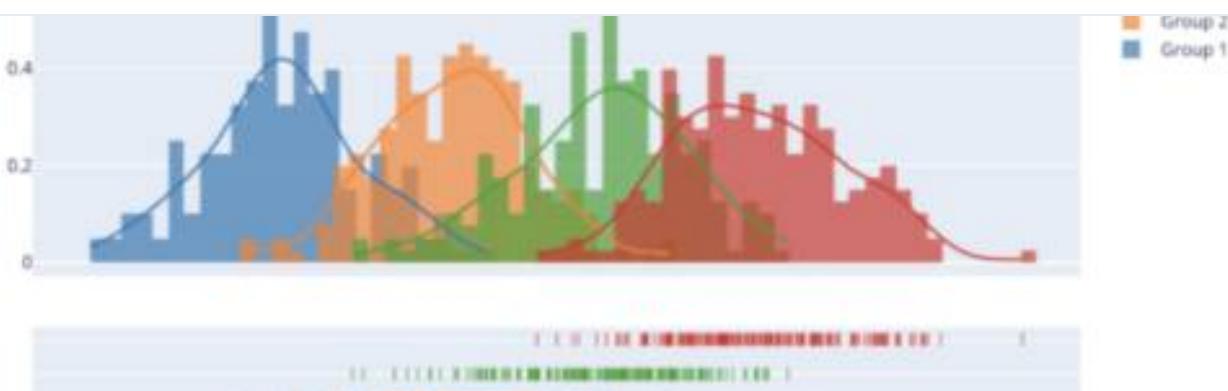
```
st.altair_chart(my_altair_chart)
```



## Vega-Lite

Display a chart using the Vega-Lite library.

```
st.vega_lite_chart(my_vega_lite_chart)
```



## Plotly

Display an interactive Plotly chart.

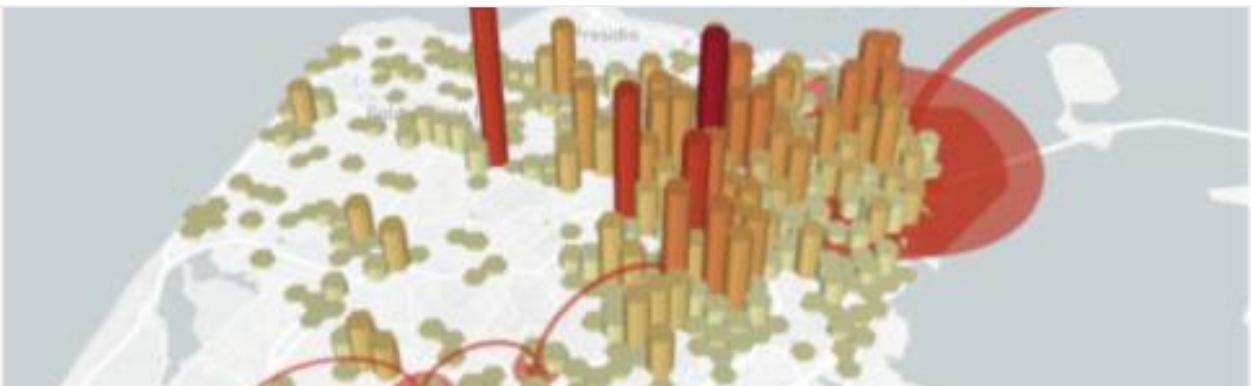
```
st.plotly_chart(my_plotly_chart)
```



## Bokeh

Display an interactive Bokeh chart.

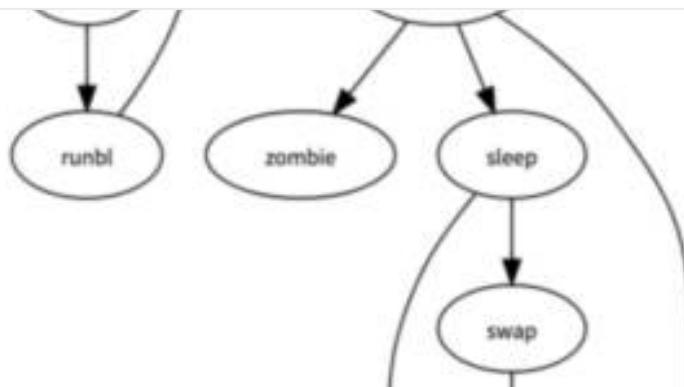
```
st.bokeh_chart(my_bokeh_chart)
```



## PyDeck

Display a chart using the PyDeck library.

```
st.pydeck_chart(my_pydeck_chart)
```



## GraphViz

Display a graph using the dagre-d3 library.

```
st.graphviz_chart(my_graphviz_spec)
```

Was this page helpful?

Yes    No

[Suggest edits](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: Data display elements](#)

[Next: st.line\\_chart](#) →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# Cheat Sheet

This is a summary of the docs, as of [Streamlit v1.8.0](#).

## Install & Import

```
streamlit run first_app.py
```

```
# Import convention
>>> import streamlit as st
```

## Command line

```
$ streamlit --help
$ streamlit run your_script.py
$ streamlit hello
$ streamlit config show
$ streamlit cache clear
$ streamlit docs
$ streamlit --version
```

## Pre-release features

```
pip uninstall streamlit
pip install streamlit-nightly --upgrade
```

[Learn more about beta and experimental features](#)

## Magic commands

```
# Magic commands implicitly
# call st.write().
'_This_ is some **Markdown***'
my_variable
'dataframe:', my_data_frame
```

## Display text

```
st.text('Fixed width text')
st.markdown('_Markdown_') # see *
st.latex(r''' e^{i\pi} + 1 = 0 ''')
st.write('Most objects') # df, err, func, keras!
st.write(['st', 'is <', 3]) # see *
st.title('My title')
st.header('My header')
st.subheader('My sub')
st.code('for i in range(8): foo()')
* optional kwarg unsafe_allow_html = True
```

## Display data

```
st.dataframe(my_dataframe)
st.table(data.iloc[0:10])
st.json({'foo':'bar', 'fu':'ba'})
st.metric('My metric', 42, 2)
```

## Display media

```
st.image('./header.png')
st.audio(data)
st.video(data)
```

## Add widgets to sidebar

```
# Just add it after st.sidebar:
>>> a = st.sidebar.radio('Select one:', [1, 2])

# Or use "with" notation:
>>> with st.sidebar:
>>>     st.radio('Select one:', [1, 2])
```

## Columns

```
# Two equal columns:  
>>> col1, col2 = st.columns(2)  
>>> col1.write("This is column 1")  
>>> col2.write("This is column 2")  
  
# Three different columns:  
>>> col1, col2, col3 = st.columns([3, 1, 1])  
# col1 is larger.  
  
# You can also use "with" notation:  
>>> with col1:  
>>>     st.radio('Select one:', [1, 2])
```

## Control flow

```
# Stop execution immediately:  
st.stop()  
  
# Rerun script immediately:  
st.experimental_rerun()  
  
# Group multiple widgets:  
>>> with st.form(key='my_form'):  
>>>     username = st.text_input('Username')  
>>>     password = st.text_input('Password')  
>>>     st.form_submit_button('Login')
```

## Display interactive widgets

```
st.button('Click me')
st.checkbox('I agree')
st.radio('Pick one', ['cats', 'dogs'])
st.selectbox('Pick one', ['cats', 'dogs'])
st.multiselect('Buy', ['milk', 'apples', 'potatoes'])
st.slider('Pick a number', 0, 100)
st.select_slider('Pick a size', ['S', 'M', 'L'])
st.text_input('First name')
st.number_input('Pick a number', 0, 10)
st.text_area('Text to translate')
st.date_input('Your birthday')
st.time_input('Meeting time')
st.file_uploader('Upload a CSV')
st.download_button('Download file', data)
st.camera_input("Take a picture")
st.color_picker('Pick a color')
```

# Use widgets' returned values in variables:

```
>>> for i in range(int(st.number_input('Num:'))):
>>>     foo()
>>>     if st.sidebar.selectbox('I:', ['f']) == 'f':
>>>         b()
>>>     my_slider_val = st.slider('Quinn Mallory', 1, 88)
>>>     st.write(my_slider_val)
```

# Disable widgets to remove interactivity:

```
>>> st.slider('Pick a number', 0, 100, disabled=True)
```

## Mutate data

```
# Add rows to a dataframe after
# showing it.

>>> element = st.dataframe(df1)
>>> element.add_rows(df2)

# Add rows to a chart after
# showing it.

>>> element = st.line_chart(df1)
>>> element.add_rows(df2)
```

## Display code

```
>>> with st.echo():
>>>     st.write('Code will be executed and printed')
```

## Placeholders, help, and options

```
# Replace any single element.

>>> element = st.empty()
>>> element.line_chart(...)
>>> element.text_input(...) # Replaces previous.

# Insert out of order.

>>> elements = st.container()
>>> elements.line_chart(...)
>>> st.write("Hello")
>>> elements.text_input(...) # Appears above "Hello".
```

```
st.help(pandas.DataFrame)
st.get_option(key)
st.set_option(key, value)
st.set_page_config(layout='wide')
st.experimental_show(objects)
st.experimental_get_query_params()
st.experimental_set_query_params(**params)
```

# Optimize performance

## Legacy caching

```
>>> @st.cache
... def foo(bar):
...     # Do something expensive in here...
...     return data
>>> # Executes foo
>>> d1 = foo(ref1)
>>> # Does not execute foo
>>> # Returns cached item by reference, d1 == d2
>>> d2 = foo(ref1)
>>> # Different arg, so function foo executes
>>> d3 = foo(ref2)
```

## Cache data objects

```
# E.g. Dataframe computation, storing downloaded data, etc.
>>> @st.experimental_memo
... def foo(bar):
...     # Do something expensive and return data
...     return data
# Executes foo
>>> d1 = foo(ref1)
# Does not execute foo
# Returns cached item by value, d1 == d2
>>> d2 = foo(ref1)
# Different arg, so function foo executes
>>> d3 = foo(ref2)
# Clear all cached entries for this function
>>> foo.clear()
# Clear values from *all* memoized functions
>>> st.experimental_memo.clear()
```

## Cache non-data objects

```
# E.g. TensorFlow session, database connection, etc.
>>> @st.experimental_singleton
... def foo(bar):
...     # Create and return a non-data object
```

```
...     return session

# Executes foo
>>> s1 = foo(ref1)

# Does not execute foo
# Returns cached item by reference, d1 == d2
>>> s2 = foo(ref1)

# Different arg, so function foo executes
>>> s3 = foo(ref2)

# Clear all cached entries for this function
>>> foo.clear()

# Clear all singleton caches
>>> st.experimental_singleton.clear()
```

## Display progress and status

```
>>> with st.spinner(text='In progress'):
>>>     time.sleep(5)
>>>     st.success('Done')
```

```
st.progress(progress_variable_1_to_100)
```

```
st.balloons()
```

```
st.snow()
```

```
st.error('Error message')
```

```
st.warning('Warning message')
```

```
st.info('Info message')
```

```
st.success('Success message')
```

```
st.exception(e)
```

Was this page helpful?

Yes

No

 Suggest edits

---

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[← Previous: Changelog](#)

[Next: Streamlit Cloud →](#)

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# My Component seems to be blinking/stuttering...how do I fix that?

Currently, no automatic debouncing of Component updates is performed within Streamlit. The Component creator themselves can decide to rate-limit the updates they send back to Streamlit.

Was this page helpful?

Yes    No

[Suggest edits](#)

---

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

[← Previous:](#) How do I add a Component to the sidebar?

**Next:** How do Streamlit Components differ from functionality provided in the base Streamlit package? [→](#)





# Components API Reference

The first step in developing a Streamlit Component is deciding whether to create a static component (i.e. rendered once, controlled by Python) or to create a bi-directional component that can communicate from Python to JavaScript and back.

## Create a static component

If your goal in creating a Streamlit Component is solely to display HTML code or render a chart from a Python visualization library, Streamlit provides two methods that greatly simplify the process: `components.html()` and `components.iframe()`.

If you are unsure whether you need bi-directional communication, **start here first!**

## Render an HTML string

While `st.text`, `st.markdown` and `st.write` make it easy to write text to a Streamlit app, sometimes you'd rather implement a custom piece of HTML. Similarly, while Streamlit natively supports many charting libraries, you may want to implement a specific HTML/JavaScript template for a new charting library. `components.html` works by giving you the ability to embed an iframe inside of a Streamlit app that contains your desired output.

## st.components.v1.html

**v1.9.0**

Display an HTML string in an iframe.

### Function signature

```
st.components.v1.html(html, width=None, height=None, scrolling=False)
```

### Parameters

`html(str)`

The HTML string to embed in the iframe.

**width** (*int*)

The width of the frame in CSS pixels. Defaults to the app's default element width.

**height** (*int*)

The height of the frame in CSS pixels. Defaults to 150.

**scrolling** (*bool*)

If True, show a scrollbar when the content is larger than the iframe. Otherwise, do not show a scrollbar.  
Defaults to False.

## Example

```
import streamlit as st
import streamlit.components.v1 as components

# bootstrap 4 collapse example
components.html(
    """
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5Bf9tH/B/Kd弯
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6a3GqjaVByc22aL5Y061NGaWzzy0c
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U024aKX2L8P04k6E6f7CATtCzIa59Ft8FkxgZF8TF" crossorigin="anonymou
    <div id="accordion">
        <div class="card">
            <div class="card-header" id="headingOne">
                <h5 class="mb-0">
                    <button class="btn btn-link" data-toggle="collapse" data-target="#collapseOne">
                        Collapsible Group Item #1
                    </button>
                </h5>
            </div>
            <div id="collapseOne" class="collapse show" aria-labelledby="headingOne">
                Collapsible Group Item #1 content
            </div>
        </div>
    </div>

```

```
<div class="card-body">
    Collapsible Group Item #1 content
</div>
</div>
<div class="card">
    <div class="card-header" id="headingTwo">
        <h5 class="mb-0">
            <button class="btn btn-link collapsed" data-toggle="collapse" data-
                Collapsible Group Item #2
            </button>
        </h5>
    </div>
    <div id="collapseTwo" class="collapse" aria-labelledby="headingTwo" data-
        <div class="card-body">
            Collapsible Group Item #2 content
        </div>
    </div>
</div>
""",  
height=600,  
)
```

## Render an iframe URL

`componentsiframe` is similar in features to `components.html`, with the difference being that `componentsiframe` takes a URL as its input. This is used for situations where you want to include an entire page within a Streamlit app.

# st.components.v1iframe

**v1.9.0**

Load a remote URL in an iframe.

### Function signature

```
st.components.v1iframe(src, width=None, height=None, scrolling=False)
```

## Parameters

### src (str)

The URL of the page to embed.

### width (int)

The width of the frame in CSS pixels. Defaults to the app's default element width.

### height (int)

The height of the frame in CSS pixels. Defaults to 150.

### scrolling (bool)

If True, show a scrollbar when the content is larger than the iframe. Otherwise, do not show a scrollbar.  
Defaults to False.

## Example

```
import streamlit as st
import streamlit.components.v1 as components

# embed streamlit docs in a streamlit app
components.iframe("https://docs.streamlit.io/en/latest")
```



## Create a bi-directional component

A bi-directional Streamlit Component has two parts:

1. A **frontend**, which is built out of HTML and any other web tech you like (JavaScript, React, Vue, etc.), and gets rendered in Streamlit apps via an iframe tag.
2. A **Python API**, which Streamlit apps use to instantiate and talk to that frontend

To make the process of creating bi-directional Streamlit Components easier, we've created a React template and a TypeScript-only template in the [Streamlit Component-template GitHub repo](#). We also provide some [example Components](#) in the same repo.

## Development Environment Setup

To build a Streamlit Component, you need the following installed in your development environment:

- Python 3.7 - Python 3.10
- Streamlit 0.63+
- [nodejs](#)
- [npm](#) or [yarn](#)

Clone the [component-template GitHub repo](#), then decide whether you want to use the React.js ("template") or plain TypeScript ("template-reactless") template.

1. Initialize and build the component template frontend from the terminal:

```
# React template
$ template/my_component/frontend
$ npm install      # Initialize the project and install npm dependencies
$ npm run start    # Start the Webpack dev server

# or

# TypeScript-only template
$ template-reactless/my_component/frontend
$ npm install      # Initialize the project and install npm dependencies
$ npm run start    # Start the Webpack dev server
```

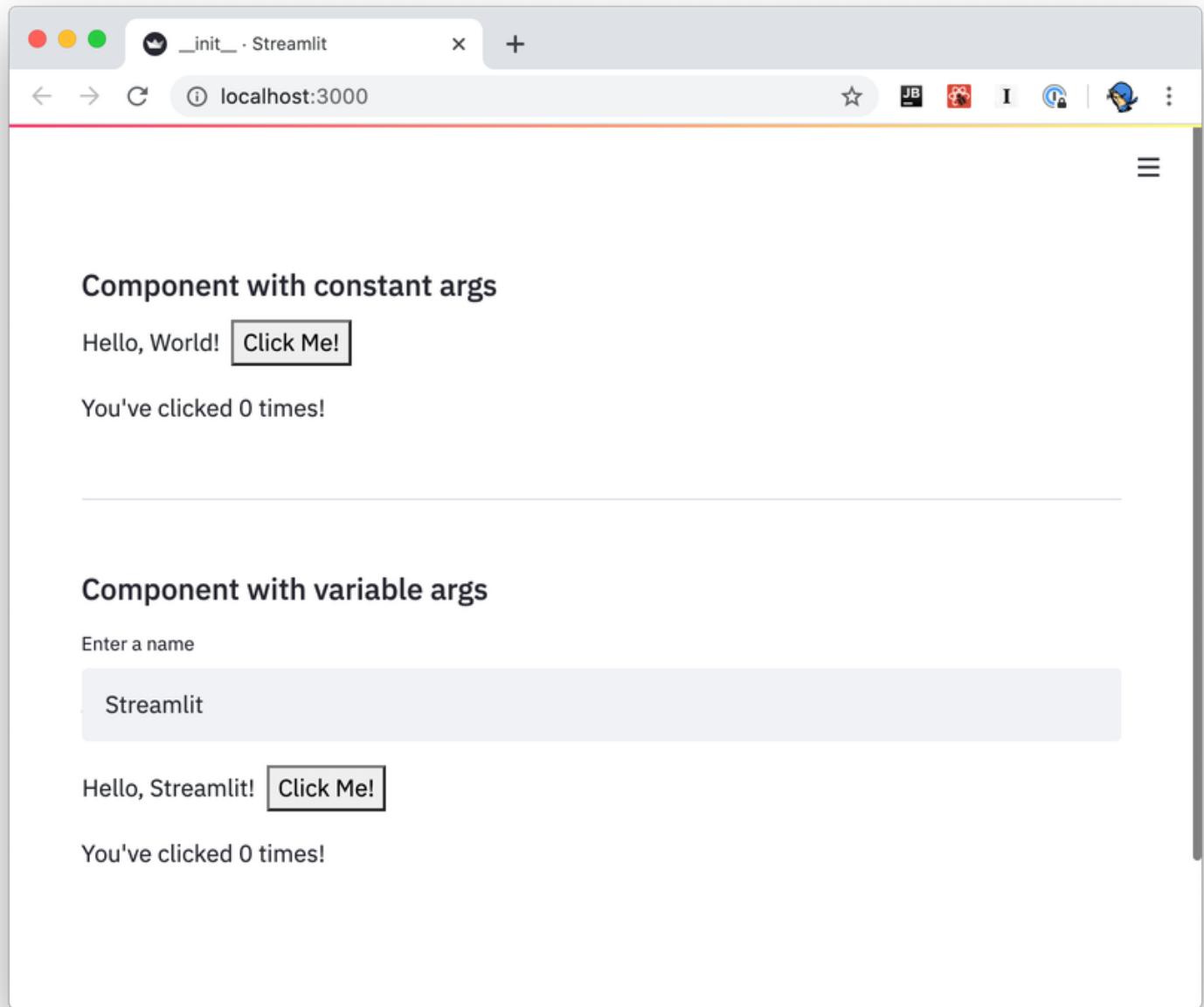
2. From a separate terminal, run the Streamlit app (Python) that declares and uses the component:

```
# React template
$ cd template
$ . venv/bin/activate # or similar to activate the venv/conda environment w/
$ streamlit run my_component/__init__.py # run the example

# or
```

```
# TypeScript-only template
$ cd template-reactless
$ . venv/bin/activate # or similar to activate the venv/conda environment w/
$ streamlit run my_component/__init__.py # run the example
```

After running the steps above, you should see a Streamlit app in your browser that looks like this:



The example app from the template shows how bi-directional communication is implemented. The Streamlit Component displays a button ([Python → JavaScript](#)), and the end-user can click the button. Each time the button is clicked, the JavaScript front-end increments the counter value and passes it back to Python ([JavaScript → Python](#)), which is then displayed by Streamlit ([Python → JavaScript](#)).

Because each Streamlit Component is its own webpage that gets rendered into an `iframe`, you can use just about any web tech you'd like to create that web page. We provide two templates to get started with in the Streamlit [Components-template GitHub repo](#); one of those templates uses [React](#) and the other does not.

## ★ Note

Even if you're not already familiar with React, you may still want to check out the React-based template. It handles most of the boilerplate required to send and receive data from Streamlit, and you can learn the bits of React you need as you go.

If you'd rather not use React, please read this section anyway! It explains the fundamentals of Streamlit ↔ Component communication.

## React

The React-based template is in `template/my_component/frontend/src/MyComponent.tsx`.

- `MyComponent.render()` is called automatically when the component needs to be re-rendered (just like in any React app)
- Arguments passed from the Python script are available via the `this.props.args` dictionary:

```
# Send arguments in Python:  
result = my_component(greeting="Hello", name="Streamlit")
```

```
// Receive arguments in frontend:  
let greeting = this.props.args["greeting"]; // name = "Hello"  
let name = this.props.args["name"]; // greeting = "Streamlit"
```

- Use `Streamlit.setComponentValue()` to return data from the component to the Python script:

```
// Set value in frontend:  
Streamlit.setComponentValue(3.14);
```

```
# Access value in Python:  
result = my_component(greeting="Hello", name="Streamlit")  
st.write("result = ", result) # result = 3.14
```



When you call `Streamlit.setComponentValue(new_value)`, that new value is sent to Streamlit, which then *re-executes the Python script from top to bottom*. When the script is re-executed, the call to `my_component(...)` will return the new value.

From a *code flow* perspective, it appears that you're transmitting data synchronously with the frontend: Python sends the arguments to JavaScript, and JavaScript returns a value to Python, all in a single function call! But in reality this is all happening *asynchronously*, and it's the re-execution of the Python script that achieves the sleight of hand.

- Use `Streamlit.setFrameHeight()` to control the height of your component. By default, the React template calls this automatically (see `StreamlitComponentBase.componentDidUpdate()`). You can override this behavior if you need more control.
- There's a tiny bit of magic in the last line of the file: `export default withStreamlitConnection(MyComponent)` - this does some handshaking with Streamlit, and sets up the mechanisms for bi-directional data communication.

## TypeScript-only

The TypeScript-only template is in `template-reactless/my_component/frontend/src/MyComponent.tsx`.

This template has much more code than its React sibling, in that all the mechanics of handshaking, setting up event listeners, and updating the component's frame height are done manually. The React version of the template handles most of these details automatically.

- Towards the bottom of the source file, the template calls `Streamlit.setComponentReady()` to tell Streamlit it's ready to start receiving data. (You'll generally want to do this after creating and loading everything that the Component relies on.)
- It subscribes to `Streamlit.RENDER_EVENT` to be notified of when to redraw. (This event won't be fired until `setComponentReady` is called)
- Within its `onRender` event handler, it accesses the arguments passed in the Python script via `event.detail.args`
- It sends data back to the Python script in the same way that the React template does—clicking on the "Click Me!" button calls `Streamlit.setComponentValue()`
- It informs Streamlit when its height may have changed via `Streamlit.setFrameHeight()`

# Working with Themes

## Note

Custom component theme support requires streamlit-component-lib version 1.2.0 or higher.

Along with sending an `args` object to your component, Streamlit also sends a `theme` object defining the active theme so that your component can adjust its styling in a compatible way. This object is sent in the same message as `args`, so it can be accessed via `this.props.theme` (when using the React template) or `event.detail.theme` (when using the plain TypeScript template).

The `theme` object has the following shape:

```
{  
  "base": "lightOrDark",  
  "primaryColor": "someColor1",  
  "backgroundColor": "someColor2",  
  "secondaryBackgroundColor": "someColor3",  
  "textColor": "someColor4",  
  "font": "someFont"  
}
```

The `base` option allows you to specify a preset Streamlit theme that your custom theme inherits from. Any theme config options not defined in your theme settings have their values set to those of the base theme. Valid values for `base` are `"light"` and `"dark"`.

Note that the theme object has fields with the same names and semantics as the options in the "theme" section of the config options printed with the command `streamlit config show`.

When using the React template, the following CSS variables are also set automatically.

```
--base  
--primary-color  
--background-color  
--secondary-background-color  
--text-color  
--font
```

If you're not familiar with [CSS variables](#), the TLDR version is that you can use them like this:

```
.mySelector {  
    color: var(--text-color);  
}
```



These variables match the fields defined in the [theme](#) object above, and whether to use CSS variables or the theme object in your component is a matter of personal preference.

## Other frontend details

- Because you're hosting your component from a dev server (via [npm run start](#)), any changes you make should be automatically reflected in the Streamlit app when you save.
- If you want to add more packages to your component, run [npm add](#) to add them from within your component's [frontend/](#) directory.

```
npm add baseui
```



- To build a static version of your component, run [npm run build](#). See [Prepare your Component](#) for more information



## Python API

[components.declare\\_component\(\)](#) is all that's required to create your Component's Python API:

```
import streamlit.components.v1 as components  
my_component = components.declare_component(  
    "my_component",  
    url="http://localhost:3001"  
)
```



You can then use the returned [my\\_component](#) function to send and receive data with your frontend code:

```
# Send data to the frontend using named arguments.  
return_value = my_component(name="Blackbeard", ship="Queen Anne's Revenge")  
  
# `my_component`'s return value is the data returned from the frontend.  
st.write("Value = ", return_value)
```

While the above is all you need to define from the Python side to have a working Component, we recommend creating a "wrapper" function with named arguments and default values, input validation and so on. This will make it easier for end-users to understand what data values your function accepts and allows for defining helpful docstrings.

Please see [this example](#) from the Components-template for an example of creating a wrapper function.

## Data serialization

### Python → Frontend

You send data from Python to the frontend by passing keyword args to your Component's invoke function (that is, the function returned from `declare_component`). You can send the following types of data from Python to the frontend:

- Any JSON-serializable data
- `numpy.array`
- `pandas.DataFrame`

Any JSON-serializable data gets serialized to a JSON string, and deserialized to its JavaScript equivalent.

`numpy.array` and `pandas.DataFrame` get serialized using [Apache Arrow](#) and are deserialized as instances of `ArrowTable`, which is a custom type that wraps Arrow structures and provides a convenient API on top of them.

Check out the [CustomDataframe](#) and [SelectableDataTable](#) Component example code for more context on how to use `ArrowTable`.

### Frontend → Python

You send data from the frontend to Python via the `Streamlit.setComponentValue()` API (which is part of the template code). Unlike arg-passing from Python → frontend, **this API takes a single value**. If you want to return multiple values, you'll need to wrap them in an `Array` or `Object`.

Custom Components can send JSON-serializable data from the frontend to Python, as well as [Apache Arrow ArrowTable](#) s to represent dataframes.

## Was this page helpful?

 Yes

 No

 [Suggest edits](#)

---

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[← Previous: Components](#)

[Next: Create a Component →](#)

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# Streamlit configuration

## Command-line interface

When you install Streamlit, a command-line (CLI) tool gets installed as well. The purpose of this tool is to run Streamlit apps, change Streamlit configuration options, and help you diagnose and fix issues.

To see all of the supported commands:

```
streamlit --help
```



## Run Streamlit apps

```
streamlit run your_script.py [-- script args]
```



Runs your app. At any time you can stop the server with **Ctrl+c**.

### ★ Note

When passing your script some custom arguments, **they must be passed after two dashes**. Otherwise the arguments get interpreted as arguments to Streamlit itself.

To see the Streamlit 'Hello, World!' example app, run `streamlit hello`.

## View documentation



Opens the Streamlit documentation (i.e. this website) in a web browser.

## Clear cache

streamlit cache clear



Clears persisted files from the on-disk [Streamlit cache](#), if present.

## Set configuration options

Streamlit provides four different ways to set configuration options. This list is in reverse order of precedence, i.e. command line flags take precedence over environment variables when the same configuration option is provided multiple times.

1. In a **global config file** at `~/.streamlit/config.toml` for macOS/Linux or `%userprofile%/.streamlit/config.toml` for Windows:

```
[server]
port = 80
```



2. In a **per-project config file** at `$CWD/.streamlit/config.toml`, where `$CWD` is the folder you're running Streamlit from.

3. Through **STREAMLIT\_\*** **environment variables**, such as:

```
export STREAMLIT_SERVER_PORT=80
export STREAMLIT_SERVER_COOKIE_SECRET=dontforgottochangeme
```



4. As **flags on the command line** when running `streamlit run`:

```
streamlit run your_script.py --server.port 80
```



## View all configuration options

```
streamlit config show
```



Shows all config options available for Streamlit, including their current values:

```
# Streamlit version: 1.9.0
```



```
[global]
```

```
# By default, Streamlit checks if the Python watchdog module is available and
# If you'd like to turn off this warning, set this to True.
# Default: false
disableWatchdogWarning = false
```

```
# If True, will show a warning when you run a Streamlit-enabled script via "py
# Default: true
showWarningOnDirectExecution = true
```

```
# DataFrame serialization.
```

```
# Acceptable values: - 'legacy': Serialize DataFrames using Streamlit's custom
# Default: "arrow"
dataFrameSerialization = "arrow"
```

```
[logger]
```

```
# Level of logging: 'error', 'warning', 'info', or 'debug'.
# Default: 'info'
level = "info"
```

```
# String format for logging messages. If logger.datetimeFormat is set, logger
# Default: "%(asctime)s %(message)s"
messageFormat = "%(asctime)s %(message)s"
```

[client]

```
# Whether to enable st.cache.  
# Default: true  
caching = true  
  
# If false, makes your Streamlit script not draw to a Streamlit app.  
# Default: true  
displayEnabled = true  
  
# Controls whether uncaught app exceptions are displayed in the browser. By default, they are.  
# If set to False, an exception will result in a generic message being shown :  
# Default: true  
showErrorDetails = true
```

[runner]

```
# Allows you to type a variable or string by itself in a single line of Python code.  
# Default: true  
magicEnabled = true  
  
# Install a Python tracer to allow you to stop or pause your script at any point.  
# Default: false  
installTracer = false  
  
# Sets the MPLBACKEND environment variable to Agg inside Streamlit to prevent Matplotlib from using Tk.  
# Default: true  
fixMatplotlib = true  
  
# Run the Python Garbage Collector after each script execution. This can help free up memory.  
# Default: true  
postScriptGC = true  
  
# Handle script rerun requests immediately, rather than waiting for script execution to finish.  
# Default: false  
fastReruns = false
```

[server]

```
# List of folders that should not be watched for changes. This impacts both "I
# Relative paths will be taken as relative to the current working directory.
# Example: ['/home/user1/env', 'relative/path/to/folder']
# Default: []
folderWatchBlacklist = []

# Change the type of file watcher used by Streamlit, or turn it off completely
# Allowed values: * "auto" : Streamlit will attempt to use the watchdog module
# Default: "auto"
fileWatcherType = "auto"

# Symmetric key used to produce signed cookies. If deploying on multiple repl
# Default: randomly generated secret key.
cookieSecret =

# If false, will attempt to open a browser window on start.
# Default: false unless (1) we are on a Linux box where DISPLAY is unset, or
headless = false

# Automatically rerun script when the file is modified on disk.
# Default: false
runOnSave = false

# The address where the server will listen for client and browser connections.
# Default: (unset)
# address =

# The port where the server will listen for browser connections.
# Default: 8501
port = 8501

# The base path for the URL where Streamlit should be served from.
# Default: ""
baseUrlPath = ""

# Enables support for Cross-Origin Request Sharing (CORS) protection, for added
# Due to conflicts between CORS and XSRF, if `server.enableXsrfProtection` is
# Default: true
enableCORS = true

# Enables support for Cross-Site Request Forgery (XSRF) protection, for added
# Due to conflicts between CORS and XSRF, if `server.enableXsrfProtection` is
# Default: true
```

```
enableXsrfProtection = true

# Max size, in megabytes, for files uploaded with the file_uploader.
# Default: 200
maxUploadSize = 200

# Max size, in megabytes, of messages that can be sent via the WebSocket connection.
# Default: 200
maxMessageSize = 200

# Enables support for websocket compression.
# Default: false
enableWebsocketCompression = false

[browser]

# Internet address where users should point their browsers in order to connect to Streamlit.
# This is used to: - Set the correct URL for CORS and XSRF protection purposes.
# Default: 'localhost'
serverAddress = "localhost"

# Whether to send usage statistics to Streamlit.
# Default: true
gatherUsageStats = true

# Port where users should point their browsers in order to connect to the application.
# This is used to: - Set the correct URL for CORS and XSRF protection purposes.
# Default: whatever value is set in server.port.
serverPort = 8501

[mapbox]

# Configure Streamlit to use a custom Mapbox token for elements like st.pydeck_map.
# Default: ""
token = ""

[deprecation]

# Set to false to disable the deprecation warning for the file uploader encoding.
# Default: true
```

```
showfileUploaderEncoding = true

# Set to false to disable the deprecation warning for using the global pyplot
# Default: true
showPyplotGlobalUse = true

[theme]

# The preset Streamlit theme that your custom theme inherits from. One of "light",
# "base", or "dark".
# base =

# Primary accent color for interactive elements.
# primaryColor =

# Background color for the main content area.
# backgroundColor =

# Background color used for the sidebar and most interactive widgets.
# secondaryBackgroundColor =

# Color used for almost all text.
# textColor =

# Font family for all text in the app, except code blocks. One of "sans serif",
# "serif", or "mono".
# font =
```

## Was this page helpful?

 Yes     No

 [Suggest edits](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

**Next:** Optimize performance with st.cache →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# Control flow

## Change execution

By default, Streamlit apps execute the script entirely, but we allow some functionality to handle control flow in your applications.

---

Stops execution  
immediately.

Stop  
execution

`st.stop()`

---

Rerun the script  
immediately.

Rerun script

`st.experimental_rerun()`

## Group multiple widgets

By default, Streamlit reruns your script everytime a user interacts with your app. However, sometimes it's a better user experience to wait until a group of related widgets is filled before actually rerunning the script. That's what `st.form` is for!

---

Create a form  
that batches  
elements  
together with a  
“Submit” button.

Forms

`with st.form("Form"):`  
    `username = st.text_input("Username")`  
    `password = st.text_input("Password")`  
    `submit_button = st.form_submit_button("Login")`

---

Display a form  
submit button.

Form submit  
button

with st.  
username  
password  
st.form

## Was this page helpful?

 Yes     No

 [Suggest edits](#)

---

## Still have questions?

 Our [forums](#) are full of helpful information and Streamlit experts.

[← Previous: Status elements](#)

[Next: st.stop →](#)

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



- [Home/](#)
- [Streamlit library/](#)
- [Get started/](#)
- [Create an app](#)

1. Contents
2. [Create an app](#)
3. [Create your first app](#)
4. [Fetch some data](#)
5. [Effortless caching](#)
6. [How's it work?](#)
7. [Inspect the raw data](#)
8. [Draw a histogram](#)
9. [Plot data on a map](#)
10. [Filter results with a slider](#)
11. [Use a button to toggle data](#)
12. [Let's put it all together](#)
13. [Share your app](#)
14. [Get help](#)

## Create an app

If you've made it this far, chances are you've [installed Streamlit](#) and run through the basics in our [Main concepts](#) guide. If not, now is a good time to take a look.

The easiest way to learn how to use Streamlit is to try things out yourself. As you read through this guide, test each method. As long as your app is running, every time you add a new element to your script and save, Streamlit's UI will ask if you'd like to rerun the app and view the changes. This allows you to work in a fast interactive loop: you write some code, save it, review the output, write some more, and so on, until you're happy with the results. The goal is to use Streamlit to create an interactive app for your data or model and along the way to use Streamlit to review, debug, perfect, and share your code.

In this guide, you're going to use Streamlit's core features to create an interactive app; exploring a public Uber dataset for pickups and drop-offs in New York City. When you're finished, you'll know how to fetch and cache data, draw charts, plot information on a map, and use interactive widgets, like a slider, to filter results.

*star*    **Tip**

If you'd like to skip ahead and see everything at once, the [complete script is available below](#).

## Create your first app

Streamlit is more than just a way to make data apps, it's also a community of creators that share their apps and ideas and help each other make their work better. Please come join us on the community forum. We love to hear your questions, ideas, and help you work through your bugs — stop by today!

1. The first step is to create a new Python script. Let's call it `uber_pickups.py`.

2. Open `uber_pickups.py` in your favorite IDE or text editor, then add these lines:

```
import streamlit as st  
import pandas as pd  
import numpy as np
```

3. Every good app has a title, so let's add one:

```
st.title('Uber pickups in NYC')
```

4. Now it's time to run Streamlit from the command line:

```
streamlit run uber_pickups.py
```

Running a Streamlit app is no different than any other Python script. Whenever you need to view the app, you can use this command.

*star* **Tip**

Did you know you can also pass a URL to `streamlit run` ? This is great when combined with Github Gists. For example:

```
$ streamlit run https://raw.githubusercontent.com/streamlit/demo-uber-nyc-pickups/master/
```

5. As usual, the app should automatically open in a new tab in your browser.

## Fetch some data

Now that you have an app, the next thing you'll need to do is fetch the Uber dataset for pickups and drop-offs in New York City.

1. Let's start by writing a function to load the data. Add this code to your script:

```
DATE_COLUMN = 'date/time'
DATA_URL = ('https://s3-us-west-2.amazonaws.com/'
            'streamlit-demo-data/uber-raw-data-sep14.csv.gz')

def load_data(nrows):
    data = pd.read_csv(DATA_URL, nrows=nrows)
    lowercase = lambda x: str(x).lower()
    data.rename(lowercase, axis='columns', inplace=True)
    data[DATE_COLUMN] = pd.to_datetime(data[DATE_COLUMN])
    return data
```

You'll notice that `load_data` is a plain old function that downloads some data, puts it in a Pandas dataframe, and converts the date column from text to datetime. The function accepts a single parameter (`nrows`), which specifies the number of rows that you want to load into the dataframe.

2. Now let's test the function and review the output. Below your function, add these lines:

```
# Create a text element and let the reader know the data is loading.
data_load_state = st.text('Loading data...')

# Load 10,000 rows of data into the dataframe.
data = load_data(10000)

# Notify the reader that the data was successfully loaded.
data_load_state.text('Loading data...done!')
```

You'll see a few buttons in the upper-right corner of your app asking if you'd like to rerun the app. Choose **Always rerun**, and you'll see your changes automatically each time you save.

Ok, that's underwhelming...

It turns out that it takes a long time to download data, and load 10,000 lines into a dataframe. Converting the date column into datetime isn't a quick job either. You don't want to reload the data each time the app is updated – luckily Streamlit allows you to cache the data.

## Effortless caching

1. Try adding `@st.cache` before the `load_data` declaration:

```
@st.cache
def load_data(nrows):
```

2. Then save the script, and Streamlit will automatically rerun your app. Since this is the first time you're running the script with `@st.cache`, you won't see anything change. Let's tweak your file a little bit more so that you can see the power of caching.

3. Replace the line `data_load_state.text('Loading data...done!')` with this:

```
data_load_state.text("Done! (using st.cache)")
```



4. Now save. See how the line you added appeared immediately? If you take a step back for a second, this is actually quite amazing. Something magical is happening behind the scenes, and it only takes one line of code to activate it.

## How's it work?

Let's take a few minutes to discuss how `@st.cache` actually works.

When you mark a function with Streamlit's cache annotation, it tells Streamlit that whenever the function is called that it should check three things:

1. The actual bytecode that makes up the body of the function
2. Code, variables, and files that the function depends on
3. The input parameters that you called the function with

If this is the first time Streamlit has seen these items, with these exact values, and in this exact combination, it runs the function and stores the result in a local cache. The next time the function is called, if the three values haven't changed, then Streamlit knows it can skip executing the function altogether. Instead, it reads the output from the local cache and passes it on to the caller -- like magic.

"But, wait a second," you're saying to yourself, "this sounds too good to be true. What are the limitations of all this awesomesauce?"

Well, there are a few:

1. Streamlit will only check for changes within the current working directory. If you upgrade a Python library, Streamlit's cache will only notice this if that library is installed inside your working directory.
2. If your function is not deterministic (that is, its output depends on random numbers), or if it pulls data from an external time-varying source (for example, a live stock market ticker service) the cached value will be none-the-wiser.
3. Lastly, you should not mutate the output of a cached function since cached values are stored by reference (for performance reasons and to be able to support libraries such as TensorFlow). Note that, here, Streamlit is smart enough to detect these mutations and show a loud warning explaining how to fix the problem.

While these limitations are important to keep in mind, they tend not to be an issue a surprising amount of the time. Those times, this cache is really transformational.

### *star* **Tip**

Whenever you have a long-running computation in your code, consider refactoring it so you can use `@st.cache`, if possible.

Now that you know how caching with Streamlit works, let's get back to the Uber pickup data.

## Inspect the raw data

It's always a good idea to take a look at the raw data you're working with before you start working with it. Let's add a subheader and a printout of the raw data to the app:

```
st.subheader('Raw data')
st.write(data)
```



In the [Main concepts](#) guide you learned that `st.write` will render almost anything you pass to it. In this case, you're passing in a dataframe and it's rendering as an interactive table.

`st.write` tries to do the right thing based on the data type of the input. If it isn't doing what you expect you can use a specialized command like `st.dataframe` instead. For a full list, see [API reference](#).

## Draw a histogram

Now that you've had a chance to take a look at the dataset and observe what's available, let's take things a step further and draw a histogram to see what Uber's busiest hours are in New York City.

1. To start, let's add a subheader just below the raw data section:

```
st.subheader('Number of pickups by hour')
```



2. Use NumPy to generate a histogram that breaks down pickup times binned by hour:

```
hist_values = np.histogram(
    data[DATE_COLUMN].dt.hour, bins=24, range=(0,24))[0]
```



3. Now, let's use Streamlit's `st.bar_chart()` method to draw this histogram.

```
st.bar_chart(hist_values)
```



4. Save your script. This histogram should show up in your app right away. After a quick review, it looks like the busiest time is 17:00 (5 P.M.).

To draw this diagram we used Streamlit's native `bar_chart()` method, but it's important to know that Streamlit supports more complex charting libraries like Altair, Bokeh, Plotly, Matplotlib and more. For a full list, see [supported charting libraries](#).

## Plot data on a map

Using a histogram with Uber's dataset helped us determine what the busiest times are for pickups, but what if we wanted to figure out where pickups were concentrated throughout the city. While you could use a bar chart to show this data, it wouldn't be easy to interpret unless you were intimately familiar with latitudinal and longitudinal coordinates in the city. To show pickup concentration, let's use Streamlit `st.map()` function to overlay the data on a map of New York City.

## 1. Add a subheader for the section:

```
st.subheader('Map of all pickups')
```

## 2. Use the `st.map()` function to plot the data:

```
st.map(data)
```

## 3. Save your script. The map is fully interactive. Give it a try by panning or zooming in a bit.

After drawing your histogram, you determined that the busiest hour for Uber pickups was 17:00. Let's redraw the map to show the concentration of pickups at 17:00.

## 1. Locate the following code snippet:

```
st.subheader('Map of all pickups')
st.map(data)
```

## 2. Replace it with:

```
hour_to_filter = 17
filtered_data = data[data[DATE_COLUMN].dt.hour == hour_to_filter]
st.subheader(f'Map of all pickups at {hour_to_filter}:00')
st.map(filtered_data)
```

## 3. You should see the data update instantly.

To draw this map we used the `st.map` function that's built into Streamlit, but if you'd like to visualize complex map data, we encourage you to take a look at the [st.pydeck\\_chart](#).

## Filter results with a slider

In the last section, when you drew the map, the time used to filter results was hardcoded into the script, but what if we wanted to let a reader dynamically filter the data in real time? Using Streamlit's widgets you can. Let's add a slider to the app with the `st.slider()` method.

## 1. Locate `hour_to_filter` and replace it with this code snippet:

```
hour_to_filter = st.slider('hour', 0, 23, 17) # min: 0h, max: 23h, default: 17h
```



2. Use the slider and watch the map update in real time.

## Use a button to toggle data

Sliders are just one way to dynamically change the composition of your app. Let's use the `st.checkbox` function to add a checkbox to your app. We'll use this checkbox to show/hide the raw data table at the top of your app.

1. Locate these lines:

```
st.subheader('Raw data')
st.write(data)
```



2. Replace these lines with the following code:

```
if st.checkbox('Show raw data'):
    st.subheader('Raw data')
    st.write(data)
```



We're sure you've got your own ideas. When you're done with this tutorial, check out all the widgets that Streamlit exposes in our [API Reference](#).

## Let's put it all together

That's it, you've made it to the end. Here's the complete script for our interactive app.

*star* **Tip**

If you've skipped ahead, after you've created your script, the command to run Streamlit is `streamlit run [app name]`.

```
import streamlit as st
import pandas as pd
```



```

import numpy as np

st.title('Uber pickups in NYC')

DATE_COLUMN = 'date/time'
DATA_URL = ('https://s3-us-west-2.amazonaws.com/'
            'streamlit-demo-data/uber-raw-data-sep14.csv.gz')

@st.cache
def load_data(nrows):
    data = pd.read_csv(DATA_URL, nrows=nrows)
    lowercase = lambda x: str(x).lower()
    data.rename(lowercase, axis='columns', inplace=True)
    data[DATE_COLUMN] = pd.to_datetime(data[DATE_COLUMN])
    return data

data_load_state = st.text('Loading data...')
data = load_data(10000)
data_load_state.text("Done! (using st.cache)")

if st.checkbox('Show raw data'):
    st.subheader('Raw data')
    st.write(data)

st.subheader('Number of pickups by hour')
hist_values = np.histogram(data[DATE_COLUMN].dt.hour, bins=24, range=(0,24))[0]
st.bar_chart(hist_values)

# Some number in the range 0-23
hour_to_filter = st.slider('hour', 0, 23, 17)
filtered_data = data[data[DATE_COLUMN].dt.hour == hour_to_filter]

st.subheader('Map of all pickups at %s:00' % hour_to_filter)
st.map(filtered_data)

```

## Share your app

After you've built a Streamlit app, it's time to share it! To show it off to the world you can use **Streamlit Cloud** to deploy, manage, and share your app for free.

It works in 3 simple steps:

1. Put your app in a public GitHub repo (and make sure it has a requirements.txt!)
2. Sign into [share.streamlit.io](#)
3. Click 'Deploy an app' and then paste in your GitHub URL

That's it! 🎉 You now have a publicly deployed app that you can share with the world. Click to learn more about [how to use Streamlit Cloud](#).

## Get help

That's it for getting started, now you can go and build your own apps! If you run into difficulties here are a few things you can do.

- Check out our [community forum](#) and post a question
- Quick help from command line with `$ streamlit help`
- Go through our [Knowledge Base](#) for tips, step-by-step tutorials, and articles that answer your questions about creating and deploying Streamlit apps.
- Read more documentation! Check out:
  - [Advanced features](#) for things like caching, theming, and adding statefulness to apps.
  - [API reference](#) for examples of every Streamlit command.

## Was this page helpful?

[Yes](#)

[No](#)

[edit](#) [Suggest edits](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: Installation](#)

[Next: API reference](#) →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# Create a Component

## ★ Note

If you are only interested in **using Streamlit Components**, then you can skip this section and head over to the [Streamlit Components Gallery](#) to find and install components created by the community!

Developers can write JavaScript and HTML "components" that can be rendered in Streamlit apps. Streamlit Components can receive data from, and also send data to, Streamlit Python scripts.

Streamlit Components let you expand the functionality provided in the base Streamlit package. Use Streamlit Components to create the needed functionality for your use-case, then wrap it up in a Python package and share with the broader Streamlit community!

## Types of Streamlit Components you could create include:

- Custom versions of existing Streamlit elements and widgets, such as `st.slider` or `st.file_uploader`.
- Completely new Streamlit elements and widgets by wrapping existing React.js, Vue.js, or other JavaScript widget toolkits.
- Rendering Python objects having methods that output HTML, such as IPython `__repr_html__`.
- Convenience functions for commonly-used web features like [GitHub gists](#) and [Pastebin](#).

Check out these Streamlit Components tutorial videos by Streamlit engineer Tim Conkling to get started:

## Part 1: Setup and Architecture



Watch on  YouTube

## Part 2: Make a Slider Widget



Watch on  YouTube

Was this page helpful?

 Yes

 No

 [Suggest edits](#)

---

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[\*\*← Previous:\*\* Components API](#)

**Next:** Publish a Component →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# Data display elements

When you're working with data, it is extremely valuable to visualize that data quickly, interactively, and from multiple different angles. That's what Streamlit is actually built and optimized for.

You can display data via [charts](#), and you can display it in raw form. These are the Streamlit commands you can use to display raw data.

Country	Avg. Temp	Avg. Precip	Avg. Wind	Avg. Humidity	Avg. Clouds	Avg. Rain
Brazil	16.5646	17.3449	17.4430	17.6581	20.3672	19.2388
China	58.3407	60.6909	63.9427	68.4626	74.7479	80.4459
France	23.4085	26.6540	25.2027	25.3211	26.7809	25.4474
Germany	22.0777	24.2415	25.3408	24.8436	23.9885	25.2363
India	50.1427	50.0158	51.3677	52.2677	49.4983	49.3096
Indonesia	7.3837	8.0549	7.6509	8.0770	7.9255	8.2192
Mexico	6.6326	7.0989	7.3942	8.0449	8.6041	9.1411

Display a  
dataframe as an  
interactive table.

Dataframes

`st.datafr`

	A	B	C	D	E
0	1.000000	1.329212	nan	-0.316280	-0.990810
1	2.000000	-1.070816	-1.438713	0.564417	0.295722
2	3.000000	-1.626404	0.219565	0.678805	1.889273
3	4.000000	0.961538	0.104011	-0.481165	0.850229
4	5.000000	1.453425	1.057737	0.165562	0.515018

Display a static  
table.

Static tables

`st.table(`

Temperature

Wind

Humidity

70 °F

↑ 1.2 °F

9 mph

↓ -8%

86%

↑ 4%

Metrics

Display a metric in big bold font, with an optional indicator of how the metric changed.

st.metric

```
    "init-param" : {...}
}
{
  "servlet-name" : "cofaxCDS"
  "servlet-class" : "org.cofax.cds.CDSServlet"
  {
    "init-param" : {
      "configGlossary:installationAt" : "Philadelphia, PA"
      "configGlossary:adminEmail" : "ksm@pobox.com"
    }
  }
}
```

Dicts and JSON

Display object or string as a pretty-printed JSON string.

st.json(r

## Was this page helpful?

Yes

No

Suggest edits

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# Connect Streamlit to data sources

These step-by-step guides demonstrate how to connect Streamlit apps to various databases & APIs. They use Streamlit's [secrets management](#) and [caching](#) to provide secure and fast data access.

- [AWS S3](#)
- [BigQuery](#)
- [Snowflake](#)
- [Microsoft SQL Server](#)
- [Firestore \(blog\)](#)
- [MongoDB](#)
- [MySQL](#)
- [PostgreSQL](#)
- [Tableau](#)
- [Private Google Sheet](#)
- [Public Google Sheet](#)
- [TigerGraph](#)
- [Deta Base](#)
- [Supabase](#)
- [Google Cloud Storage](#)

**Was this page helpful?**

Yes

No



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[← Previous: Tutorials](#)

[Next: AWS S3 →](#)

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# Installing dependencies

- [ModuleNotFoundError: No module named](#)
- [ImportError: libGL.so.1: cannot open shared object file: No such file or directory](#)
- [ERROR: No matching distribution found for](#)
- [How to install a package not on PyPI/Conda but available on GitHub](#)

Was this page helpful?

Yes

No

[Suggest edits](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

[← Previous: Streamlit Components](#)

[Next: How to install a package not on PyPI or Conda but available on GitHub →](#)





How can I deploy multiple Streamlit apps on different subdomains?

# How can I deploy multiple Streamlit apps on different subdomains?

## Problem

You want to deploy multiple Streamlit apps on different subdomains.

## Solution

Like running your Streamlit app on more common ports such as 80, subdomains are handled by a web server like Apache or Nginx:

- Set up a web server on a machine with a public IP address, then use a DNS server to point all desired subdomains to your webserver's IP address
- Configure your web server to route requests for each subdomain to the different ports that your Streamlit apps are running on

For example, let's say you had two Streamlit apps called `Calvin` and `Hobbes`. App `Calvin` is running on port **8501**. You set up app `Hobbes` to run on port **8502**. Your webserver would then be set up to "listen" for requests on subdomains `calvin.somedomain.com` and `hobbes.subdomain.com`, and route requests to port **8501** and **8502**, respectively.

Check out these two tutorials for Apache2 and Nginx that deal with setting up a webserver to redirect subdomains to different ports:

- [Apache2 subdomains](#)
- [NGinx subdomains](#)

Was this page helpful?

Yes

No

 [Suggest edits](#)

---

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[← Previous:](#) Authentication without SSO

**Next:** How do I deploy Streamlit on a domain so it appears to run on a regular port (i.e. port 80)? [→](#)

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



How do I deploy Streamlit on a domain so it appears to run on a regular port (i.e. port 80)?

# How do I deploy Streamlit on a domain so it appears to run on a regular port (i.e. port 80)?

## Problem

You want to deploy a Streamlit app on a domain so it appears to run on port 80.

## Solution

- You should use a **reverse proxy** to forward requests from a webserver like Apache or Nginx to the port where your Streamlit app is running. You can accomplish this in several different ways. The simplest way is to forward all requests sent to your domain so that your Streamlit app appears as the content of your website.
- Another approach is to configure your webserver to forward requests to designated subfolders (e.g. <http://awesomestuff.net/streamlitapp>) to different Streamlit apps on the same domain, as in this example config for Nginx submitted by a Streamlit community member.

Related forum posts:

- <https://discuss.streamlit.io/t/permission-denied-in-ec2-port-80/798/3>
- <https://discuss.streamlit.io/t/how-to-use-streamlit-with-nginx/378/7>

Was this page helpful?

Yes

No

---

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

← **Previous:** How can I deploy multiple Streamlit apps on different subdomains?

**Next:** How do I deploy Streamlit on Heroku, AWS, Google Cloud, etc...? →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# How do I deploy Streamlit on Heroku, AWS, Google Cloud, etc...?

## Problem

You want to deploy your Streamlit app on a cloud service other than Streamlit Cloud.

## Solution

Here are some user-submitted tutorials for different cloud services:

- How to deploy Streamlit apps to Google App Engine, by Yuichiro Tachibana (Tsuchiya)
- How to Deploy Streamlit to a Free Amazon EC2 instance, by Rahul Agarwal
- Host Streamlit on Heroku, by Maarten Grootendorst
- Host Streamlit on Azure, by Richard Peterson
- Host Streamlit on 21YunBox, by Toby Lei

You can find guides for other hosting providers on our community-supported deployment wiki.

## Was this page helpful?

Yes

No

Suggest edits



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

← **Previous:** How do I deploy Streamlit on a domain so it appears to run on a regular port (i.e. port 80)?

**Next:** Does Streamlit support the WSGI Protocol? (aka Can I deploy Streamlit with gunicorn?) →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# Deployment-related questions and errors

- [How do I deploy Streamlit on a domain so it appears to run on a regular port \(i.e. port 80\)?](#)
- [How can I deploy multiple Streamlit apps on different subdomains?](#)
- [How do I deploy Streamlit on Heroku, AWS, Google Cloud, etc...?](#)
- [Invoking a Python subprocess in a deployed Streamlit app](#)
- [Does Streamlit support the WSGI Protocol? \(aka Can I deploy Streamlit with gunicorn?\)](#)
- [Argh. This app has gone over its resource limits.](#)
- [App is not loading when running remotely](#)
- [Authentication without SSO](#)
- [I don't have SSO. How do I sign in to Streamlit Cloud?](#)
- [How do I share apps with viewers outside my organization?](#)
- [Upgrade the Streamlit version of your app on Streamlit Cloud](#)
- [Organizing your apps with workspaces on Streamlit Cloud](#)
- [How do I increase the upload limit of st.file\\_uploader on Streamlit Cloud?](#)

Was this page helpful?

Yes    No

[Suggest edits](#)



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[← Previous: Installing dependencies](#)

[Next: Authentication without SSO →](#)

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.

# Documentation



Search



Streamlit library

- Get started



- API reference



- Advanced features



- Components



- Changelog

- Cheat sheet



Streamlit Cloud

- Get started



- Trust and Security

- Release notes

- Troubleshooting



Knowledge base

- Tutorials



- Using Streamlit

- Streamlit Components

- Installing dependencies
- Deployment issues

[Home](#) / [Knowledge base](#) / [Deployment issues](#) /

[Does Streamlit support the WSGI Protocol? \(aka Can I deploy Streamlit with gunicorn?\)](#)

# Does Streamlit support the WSGI Protocol? (aka Can I deploy Streamlit with gunicorn?)

## Problem

You're not sure whether your Streamlit app can be deployed with gunicorn.

## Solution

Streamlit does not support the WSGI protocol at this time, so deploying Streamlit with (for example) gunicorn is not currently possible. Check out this forum thread regarding deploying Streamlit in a gunicorn-like manner to see how other users have accomplished this.

## Was this page helpful?

 Yes     No

 [Suggest edits](#)



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.



## Documentation

*search*

Search

- *description*

### Streamlit library

- Get started  
*add*
- API reference  
*add*
- Advanced features  
*add*
- Components  
*add*
- Changelog
- Cheat sheet

- *cloud*

### Streamlit Cloud

- Get started  
*add*
- Trust and Security
- Release notes*open in new*
- Troubleshooting

- *school*

### Knowledge base

- Tutorials  
*add*
- Using Streamlit
- Streamlit Components
- Installing dependencies
- Deployment issues

- *Home*/

- *Streamlit Cloud*

## Welcome to Streamlit Cloud

*8*

Streamlit's Community Cloud is a workspace for you and your team to easily deploy, manage, and collaborate on your Streamlit apps. If you're just getting started and have not yet built your first Streamlit app, check out the main Get started page first. When you're ready to share it, create a Community Cloud account and you can launch your app in just a few minutes! Deploy, manage, and share your apps with the world, directly from Streamlit — all for free.

*arrow forward*

## Get started

[Learn how to set up your account to start deploying apps.](#)

*flight takeoff*

## Deploy an app

[A step by step guide on how to get your app deployed.](#)

[electrical services](#)

## Connect data sources

[Learn how to securely connect your app to data sources.](#)

[share](#)

## Share your app

[Share your app publicly or privately with select viewers and developers.](#)

[manage accounts](#)

## Manage your app

[Access logs, get more resources for your app, and other tips and tricks.](#)

[push pin](#)

## Note

Interested in our security model? Check out our [Trust and Security page](#).

Questions? Reach out to us on the [Community forum](#)!

Was this page helpful?

[thumb\\_up Yes](#) [thumb\\_down No](#)

[edit](#) [Suggest edits](#)

[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: Streamlit library](#) [Next: Get started](#) →

---

[Home](#) [Contact Us](#) [Community](#)



Copyright © 2022, Streamlit Inc.

## Documentation

*search*

Search

- *description*

### Streamlit library

- Get started  
*remove*
  - Main concepts
  - Installation
  - Create an app
- API reference  
*add*
- Advanced features  
*add*
- Components  
*add*
- Changelog
- Cheat sheet

- *cloud*

### Streamlit Cloud

- Get started  
*add*
- Trust and Security
- Release notes[open in new](#)
- Troubleshooting

- *school*

### Knowledge base

- Tutorials  
*add*
- Using Streamlit
- Streamlit Components
- Installing dependencies
- Deployment issues

- *Home/*
- *Streamlit library/*
- *Get started*

## **Get started**



This Get Started guide explains how Streamlit works, how to install Streamlit on your preferred operating system, and how to create your first Streamlit app!

### *description*

Main concepts introduces you to Streamlit's data model and development flow. You'll learn what makes Streamlit the most powerful way to build data apps, including the ability to display and style data, draw charts and maps, add interactive widgets, customize app layouts, cache computation, and define themes.

### *downloading*

[Installation](#) helps you set up your virtual environment and walks you through installing Streamlit on Windows, macOS, and Linux. Regardless of which package management tool and OS you're using, we recommend running the commands on this page in a virtual environment.

## [auto\\_awesome](#)

[Create an app](#) using Streamlit's core features to fetch and cache data, draw charts, plot information on a map, and use interactive widgets, like a slider, to filter results.

Was this page helpful?

Yes No

[edit](#)[Suggest edits](#)

[forum](#)

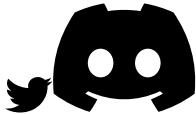
## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

←[Previous: Streamlit library](#)[Next: Main concepts](#)→

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [cloud](#)

### [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notes](#)*open in new*
- [Troubleshooting](#)

- [school](#)

### [Knowledge base](#)

- [Tutorials](#)  
*add*
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home/](#)
- [Knowledge base/](#)
- [Using Streamlit/](#)
- [How to download a file in Streamlit?](#)

## How to download a file in Streamlit?

*8*

Use the `st.download_button` widget that is natively built into Streamlit. Check out a [sample app](#) demonstrating how you can use `st.download_button` to download common file formats.

## Example usage

*8*

```
import streamlit as st

# Text files

text_contents = '''
Foo, Bar
123, 456
789, 000
'''
```

```
# Different ways to use the API

st.download_button('Download CSV', text_contents, 'text/csv')
st.download_button('Download CSV', text_contents) # Defaults to 'text/plain'

with open('myfile.csv') as f:
    st.download_button('Download CSV', f) # Defaults to 'text/plain'

# ---
# Binary files

binary_contents = b'whatever'

# Different ways to use the API

st.download_button('Download file', binary_contents) # Defaults to 'application/octet-stream'

with open('myfile.zip', 'rb') as f:
    st.download_button('Download Zip', f, file_name='archive.zip') # Defaults to 'application/octet-stream'

# You can also grab the return value of the button,
# just like with any other button.

if st.download_button(...):
    st.write('Thanks for downloading!')
```

Additional resources:

- <https://blog.streamlit.io/0-88-0-release-notes/>
- [https://share.streamlit.io/streamlit/release-demos/0.88/0.88/streamlit\\_app.py](https://share.streamlit.io/streamlit/release-demos/0.88/0.88/streamlit_app.py)
- [https://docs.streamlit.io/library/api-reference/widgets/st.download\\_button](https://docs.streamlit.io/library/api-reference/widgets/st.download_button)

Was this page helpful?

Yes No

[edit](#)[Suggest edits](#)

[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

←[Previous: How do I run my Streamlit script?](#)[Next: How to download a Pandas DataFrame as a CSV?](#)→

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.



# How to download a Pandas DataFrame as a CSV?

Use the `st.download_button` widget that is natively built into Streamlit. Check out a [sample app](#) demonstrating how you can use `st.download_button` to download common file formats.

## Example usage

```
import streamlit as st
import pandas as pd

df = pd.read_csv("dir/file.csv")

@st.cache
def convert_df(df):
    return df.to_csv().encode('utf-8')

csv = convert_df(df)

st.download_button(
    "Press to Download",
    csv,
    "file.csv",
    "text/csv",
    key='download-csv'
)
```



Additional resources:

- <https://blog.streamlit.io/0-88-0-release-notes/>

- [https://share.streamlit.io/streamlit/release-demos/0.88/0.88/streamlit\\_app.py](https://share.streamlit.io/streamlit/release-demos/0.88/0.88/streamlit_app.py)
- [https://docs.streamlit.io/library/api-reference/widgets/st.download\\_button](https://docs.streamlit.io/library/api-reference/widgets/st.download_button)

## Was this page helpful?

 Yes     No

 [Suggest edits](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

[← Previous:](#) How to download a file in Streamlit?

[Next:](#) How do I hide the hamburger menu from my app? [→](#)

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



Home / Knowledge base / Streamlit Components /

How do Streamlit Components differ from functionality provided in the base Streamlit package?

# How do Streamlit Components differ from functionality provided in the base Streamlit package?

- Streamlit Components are wrapped up in an iframe, which gives you the ability to do whatever you want (within the iframe) using any web technology you like.
- There is a strict message protocol between Components and Streamlit, which makes possible for Components to act as widgets. As Streamlit Components are wrapped in iframe, they cannot modify their parent's DOM (a.k.a the Streamlit report), which ensures that Streamlit is always secure even with user-written components.

Was this page helpful?

Yes    No

[Suggest edits](#)

---

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[Previous:](#) My Component seems to be stuttering...how do I fix that?

[Next:](#) What types of things aren't possible with Streamlit Components?

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.

## Documentation

*search*

Search

- *description*

### Streamlit library

- Get started  
*add*
- API reference  
*add*
- Advanced features  
*add*
- Components  
*add*
- Changelog
- Cheat sheet

- *cloud*

### Streamlit Cloud

- Get started  
*add*
- Trust and Security
- Release notes*open in new*
- Troubleshooting

- *school*

### Knowledge base

- Tutorials  
*add*
- Using Streamlit
- Streamlit Components
- Installing dependencies
- Deployment issues

## **Streamlit documentation**

Streamlit is an open-source Python library that makes it easy to create and share beautiful, custom web apps for machine learning and data science. In just a few minutes you can build and deploy powerful data apps. So let's get started!

*arrow forward*

### Get started

If you're new to Streamlit and don't know where to start, this is a good place.

*dvr*

### API reference

Learn about our APIs, with actionable explanations of specific functions and features.

## [App gallery](#)

[Try out awesome apps created by our users, and curated from our forums or Twitter.](#)

# How to use our docs

## description

[\*\*Streamlit library\*\*](#) includes our Get started guide, API reference, and more advanced features of the core library including caching, theming, and Streamlit Components.

## cloud

[\*\*Streamlit Cloud\*\*](#) empowers your data team to directly serve the needs of the rest of the company. Quickly go from data to app, from prototype to production. Share apps in one click and collaborate instantly with live code updates.

## school

[\*\*Knowledge base\*\*](#) is a self-serve library of tips, step-by-step tutorials, and articles that answer your questions about creating and deploying Streamlit apps.

May 17, 2022

## Leverage your user analytics on Streamlit's Community Cloud

See who viewed your apps, when, and how popular they are.

[\*\*Next:\*\* Read More →](#)

May 12, 2022

## How to share scientific analysis through a Streamlit app

3 easy steps to share your study results with fellow scientists.

[\*\*Next:\*\* Read More →](#)

May 5, 2022

## Monthly Rewind > April 2022

**Next:** [Read More →](#)

[View all updates](#)

## What's new

[photo\\_camera](#)

[st.camera\\_input](#)

[You can now upload images to your apps straight from your camera with the new st.camera\\_input feature. It's great for computer vision apps.](#)

[analytics](#)

[Workspace analytics](#)

[See how many total viewers have visited all apps in your Community Cloud workspace. Read more about how to access this feature in our docs.](#)

[group](#)

[App viewers data](#)

[Community Cloud allows you to drill down to the level of individual apps and understand their viewership better. Use app viewers data to see who has recently viewed your apps and when.](#)

[expand\\_more](#)

[Expand and collapse JSON elements with kwarg](#)

[st.json now supports a keyword-only argument, expanded, on whether the JSON should be expanded by default \(defaults to True\).](#)

[all categories](#) [all tags](#) [Categories](#) [Latest](#) [Top](#)

## Category

**Official Announcements**

Releases and other major events. Posts can be created by Streamlit employees, discussions can be participated in by all community members.

**Using Streamlit**

Post a question or issue about your Streamlit project and one of our engineers or community experts will get back to you shortly.

**Streamlit sharing**

All questions related to Streamlit sharing!

**Deploying Streamlit**

Want to share your Streamlit app on a different platform than Streamlit sharing? Ask here for help with Heroku, AWS, GCP, etc.

**Show the Community!**

Show off your Streamlit apps, Streamlit components, tutorials, code snippets, blog posts or other projects directly pertaining to Streamlit.

**Random**

New ideas for Streamlit or anything you think is interesting or pertains to data science, machine learning, data visualization, app development, etc.

## Topics

## Latest



[The Streamlit Roadmap: Big Plans for 2020!](#)

10

Mar 18



[Invite for Streamlit Sharing!](#)

31

20m



[Sidebar slider cant handle more than 8 digit inputs #33208](#)

1

27m



[Electronic Health Record Web app](#)

0

1h



[Unable to host streamlit app using heroku due to error: cannot create /app/streamlit/config.toml](#)

14

1h



[Interdependent widgets in forms](#)

2

1h



[Where are saved files?](#)

1

1h



[Different formatting when deploying in heroku versus running local](#)

0

1h

[Deploying Streamlit](#)

## Join the community

Streamlit is more than just a way to make data apps, it's also a community of creators that share their apps and ideas and help each other make their work better. Please come join us on the community forum. We love to hear your questions, ideas, and help you work through your bugs — stop by today!

[View forum](#)

## Other Media

### GitHub

[View the Streamlit source code and issue tracker.](#)

## Twitter

[Follow @streamlit on Twitter to keep up with the latest news.](#)

## YouTube

[Watch screencasts made by the Streamlit team and the community.](#)

## Discord

[Join thousands of other Streamlit enthusiasts in our Discord server.](#)

## Instagram

[Follow @streamlit.io on Instagram for more updates and content.](#)

**Next:** [Get started →](#)

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.



# How to insert elements out of order?

You can use the `st.empty` method as a placeholder, to "save" a slot in your app that you can use later.

```
st.text('This will appear first')
# Appends some text to the app.
```

```
my_slot1 = st.empty()
# Appends an empty slot to the app. We'll use this later.
```

```
my_slot2 = st.empty()
# Appends another empty slot.
```

```
st.text('This will appear last')
# Appends some more text to the app.
```

```
my_slot1.text('This will appear second')
# Replaces the first empty slot with a text string.
```

```
my_slot2.line_chart(np.random.randn(20, 2))
# Replaces the second empty slot with a chart.
```

**Was this page helpful?**

Yes    No

[Suggest edits](#)



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

← **Previous:** How do I upgrade to the latest version of Streamlit?

**Next:** What is the path of Streamlit's config.toml file? →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# Install Streamlit

## Table of contents

1. [Prerequisites](#)
2. [Install Streamlit on Windows](#)
3. [Install Streamlit on macOS/Linux](#)

## Prerequisites

Before you get started, you're going to need a few things:

- Your favorite IDE or text editor
- [Python 3.7 - Python 3.10](#)
- [PIP](#)

If you haven't already, take a few minutes to read through [Main concepts](#) to understand Streamlit's data flow model.

## Set up your virtual environment

Regardless of which package management tool you're using, we recommend running the commands on this page in a virtual environment. This ensures that the dependencies pulled in for Streamlit don't impact any other Python projects you're working on.

Below are a few tools you can use for environment management:

- [pipenv](#)
- [poetry](#)

- [venv](#)
- [virtualenv](#)
- [conda](#)

## Install Streamlit on Windows

Streamlit's officially-supported environment manager on Windows is [Anaconda Navigator](#).

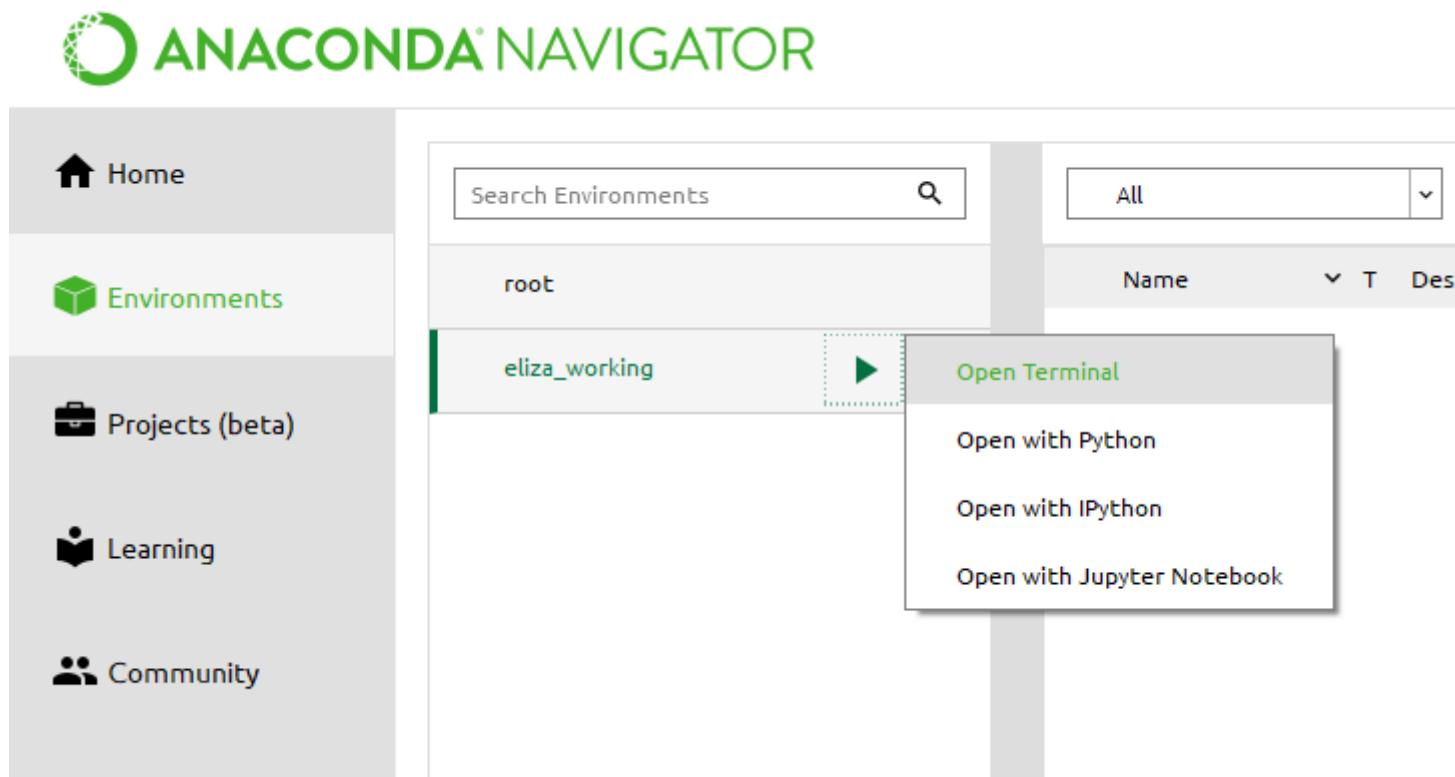
## Install Anaconda

If you don't have Anaconda install yet, follow the steps provided on the [Anaconda installation page](#).

## Create a new environment with Streamlit

Next you'll need to set up your environment.

1. Follow the steps provided by Anaconda to [set up and manage your environment](#) using the Anaconda Navigator.
2. Select the "►" icon next to your new environment. Then select "Open terminal":



3. In the terminal that appears, type:

```
pip install streamlit
```

4. Test that the installation worked:

```
streamlit hello
```

Streamlit's Hello app should appear in a new tab in your web browser!

## Use your new environment

1. In Anaconda Navigator, open a terminal in your environment (see step 2 above).
2. In the terminal that appears, use Streamlit as usual:

```
streamlit run myfile.py
```

## Install Streamlit on macOS/Linux

Streamlit's officially-supported environment manager for macOS and Linux is [Pipenv](#). See instructions on how to install and use it below.

## Install Pipenv

1. Install [pip](#).

On a macOS:

```
sudo easy_install pip
```

On Ubuntu with Python 3:

```
sudo apt-get install python3-pip
```

For other Linux distributions, see [How to install PIP for Python](#).

2. Install `pipenv`.

```
pip3 install pipenv
```

## Create a new environment with Streamlit

1. Navigate to your project folder:

```
cd myproject
```

2. Create a new Pipenv environment in that folder and activate that environment:

```
pipenv shell
```

When you run the command above, a file called `Pipfile` will appear in `myprojects/`. This file is where your Pipenv environment and its dependencies are declared.

3. Install Streamlit in your environment:

```
pip install streamlit
```

Or if you want to create an easily-reproducible environment, replace `pip` with `pipenv` every time you install something:

```
pipenv install streamlit
```

4. Test that the installation worked:

```
streamlit hello
```

Streamlit's Hello app should appear in a new tab in your web browser!

## Use your new environment

1. Any time you want to use the new environment, you first need to go to your project folder (where the [Pipenv](#) file lives) and run:

```
pipenv shell
```

2. Now you can use Python and Streamlit as usual:

```
streamlit run myfile.py
```

To stop the Streamlit server, press [ctrl-C](#).

3. When you're done using this environment, just type [exit](#) or press [ctrl-D](#) to return to your normal shell.

## Was this page helpful?

 Yes

 No

 [Suggest edits](#)



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

[\*\*← Previous:\*\* Main concepts](#)

**Next:** Create an app →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.

## Documentation

*search*

Search

- *description*

### Streamlit library

- Get started  
*add*
- API reference  
*add*
- Advanced features  
*add*
- Components  
*add*
- Changelog
- Cheat sheet

- *cloud*

### Streamlit Cloud

- Get started  
*add*
- Trust and Security
- Release notesopen in new
- Troubleshooting

- *school*

### Knowledge base

- Tutorials  
*add*
- Using Streamlit
- Streamlit Components
- Installing dependencies
- Deployment issues

- *Home/*

- *Knowledge base*

## **Knowledge base**

*8*

The knowledge base is a self-serve library of tips, step-by-step tutorials, and articles that answer your questions about creating and deploying Streamlit apps.

### *local library*

[Tutorials](#). Our tutorials include step-by-step examples of building different types of apps in Streamlit.

### *auto\_awesome*

[Using Streamlit](#). Here are some frequently asked questions about using Streamlit.

### *build*

[Streamlit Components](#). Here are some questions we've received about Streamlit Components.

[downloading](#)

[Installing dependencies](#). If you run into problems installing dependencies for your Streamlit apps, we've got you covered.

[report](#)

[Deployment issues](#). Have questions about deploying Streamlit apps to the cloud? This section covers deployment-related issues.

Was this page helpful?

Yes  No

[edit](#) [Suggest edits](#)

[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: Streamlit CloudNext: Tutorials](#) →

---

[Home](#) [Contact Us](#) [Community](#)



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*add*
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*remove*
    - [st.sidebar](#)
    - [st.columns](#)
    - [st.expander](#)
    - [st.container](#)
    - [st.empty](#)
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [\*cloud\*](#)

### [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notesopen in new](#)
- [Troubleshooting](#)

- [\*school\*](#)

## Knowledge base

- [Tutorials](#)  
*add*
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home/](#)
- [Streamlit library/](#)
- [API reference/](#)
- [Layouts and containers](#)

## Layouts and Containers

8

## Complex layouts

8

Streamlit provides several options for controlling how different elements are laid out on the screen.

**Lorem ipsum dolor sit amet**

Consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Diam quis enim lobortis scelerisque fermentum dui [faucibus in](#). Pharetra magna ac placerat vestibulum lectus mauris ultrices.

Adipiscing [elit](#) [duis tristique](#) sollicitudin. Velit aliquet sagittis id consectetur purus ut faucibus pulvinar.

**Tincidunt lobortis**

Feugiat vivamus at augue eget arcu dictum. Sed risus pretium quam vulputate. Cursus in hac habitasse platea dictumst. Aliquam ultrices sagittis orci a.

**Non diam phasellus vestibulum**

- Ut pharetra sit amet aliquam id.
- Etiam tempor orci eu lobortis elementum nibh tellus molestie nunc.
- Sed enim ut sem viverra aliquet eget sit.
- Duis at consectetur lorem donec massa sapien faucibus et molestie.

Sem integer vitae justo eget. In egestas erat imperdiet sed euismod nisi porta lorem mollis. Eu feugiat pretium nibh ipsum consequat nisl vel pretium. Elit ut aliquam purus sit amet. Aliquet nibh praesent tristique magna sit. Dapibus ultrices in iaculis nunc.

Enim eu turpis egestas pretium aenean pharetra. Nunc sed blandit libero volutpat sed cras ornare arcu. Etiam erat velit scelerisque in. Purus semper eget dui at tellus at urna condimentum mattis. Sapien

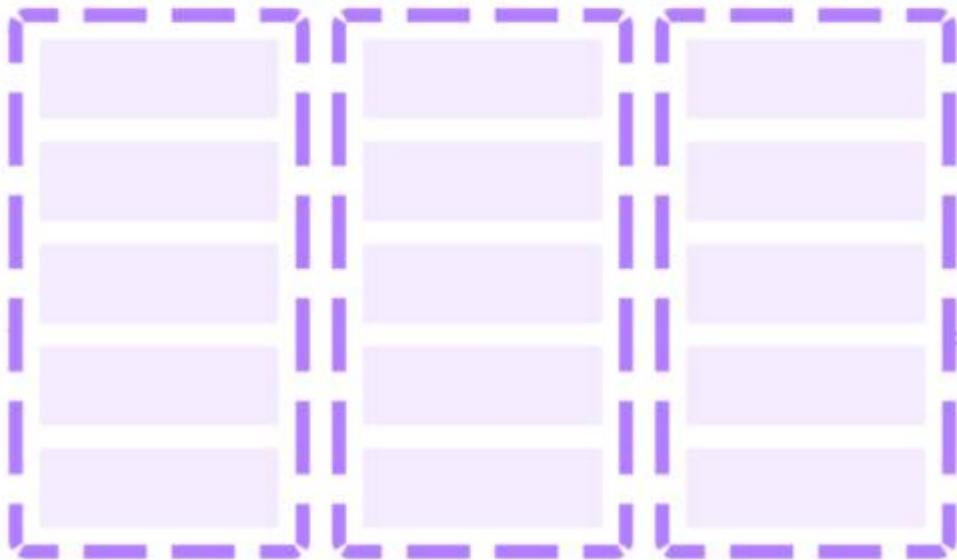
## Sidebar

Display items in a sidebar.

```
st.sidebar.write("This lives in the sidebar")
st.sidebar.button("Click me!")
```

# **Lorem ipsum dolor sit amet**

Consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. diam quis enim lobortis scelerisque fermentum dui [faucibus in](#). Pharetra magna ac placerat vestibulum lectus mauris ultrices.



Sem integer vitae justo eget. In egestas erat imperdier sed euismod nisi porta lorem mollis. Eu feugiat  
premium nibh ipsum consequat nisl vel premium. Elit ut aliquam purus sit amet. Aliquet nibh praesent  
tristique magna sit. Dapibus ultrices in iaculis nunc.

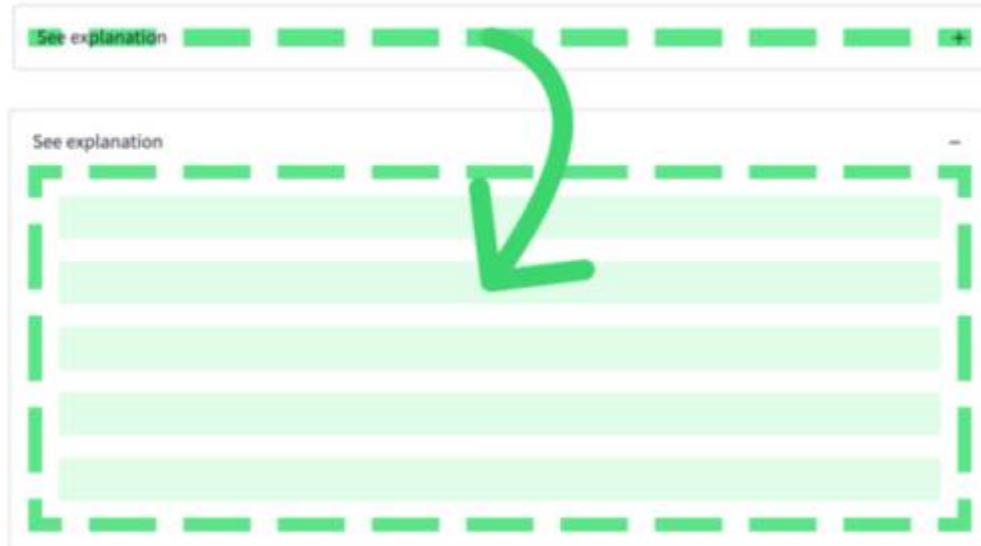
Columns

Insert containers laid out as side-by-side columns.

```
col1, col2 = st.columns(2)
col1.write("this is column 1")
col2.write("this is column 2")
```

# **Lorem ipsum dolor sit amet**

Consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. diam quis enim lobortis scelerisque fermentum dui [faucibus in](#). Pharetra magna ac placerat vestibulum lectus mauris ultrices.



Sem integer vitae justo eget. In egestas erat imperdier sed euismod nisi porta lorem mollis. Eu feugiat premium nibh ipsum consequat nisl vel premium. Elit ut aliquam purus sit amet. Aliquet nibh praesent tristique magna sit. Dapibus ultrices in iaculis nunc.

Enim eu turpis egestas pretium aenean pharetra. Nunc sed blandit libero volutpat sed cras ornare arcu. Etiam erat velit scelerisque in. Purus semper eget duis at tellus at urna condimentum mattis. Sapien feugiat et nisl sit amet facilisis sed odio euismod. Odio ut non nullam lobortis diam id.

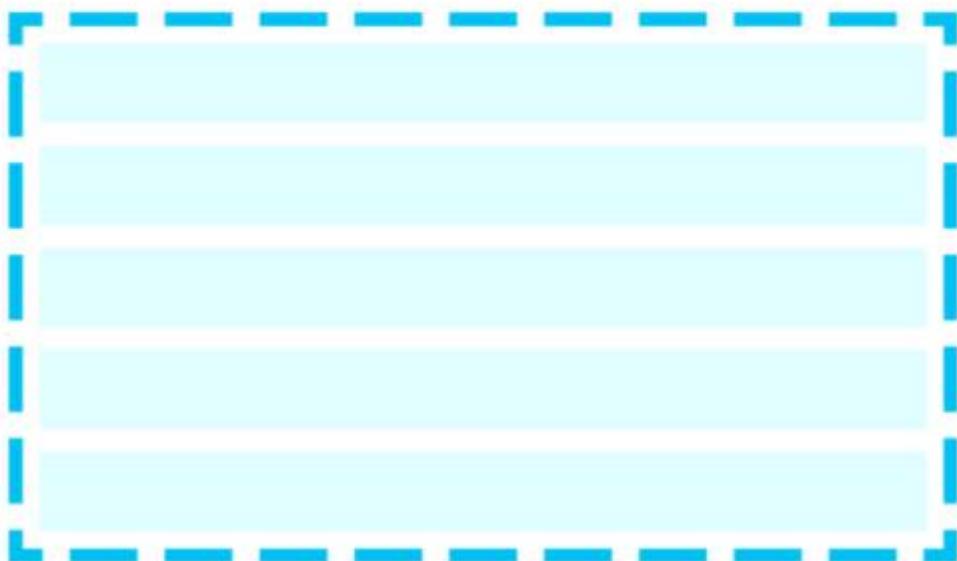
## Expander

Insert a multi-element container that can be expanded/collapsed.

```
with st.expander("Open to see more"):  
    st.write("This is more content")
```

# **Lorem ipsum dolor sit amet**

Consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. diam quis enim lobortis scelerisque fermentum dui [faucibus in](#). Pharetra magna ac placerat vestibulum lectus mauris ultrices.



Sem integer vitae justo eget. In egestas erat imperdiet sed euismod nisi porta lorem mollis. Eu feugiat pretium nibh ipsum consequat nisl vel pretium. Elit ut aliquam purus sit amet. Aliquet nibh praesent tristique magna sit. Dapibus ultrices in iaculis nunc.

Enim eu turpis egestas pretium aenean pharetra. Nunc sed blandit libero volutpat sed cras ornare arcu. Etiam erat velit scelerisque in. Purus semper eget duis at tellus at urna condimentum mattis. Sapien facilisis etiam vel felis euismod. Nullam id diam non enim. Odio non nullam pharetra donec at.

Container

Insert a multi-element container

```
c = st.container()
st.write("This will sh
c.write("This will sh
c.write("This will sh
```

# **Lorem ipsum dolor sit amet**

Consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. **Diam quis enim lobortis scelerisque fermentum dui faucibus in.** Pharetra magna ac placerat vestibulum lectus mauris ultrices.



Sem integer vitae justo eget. In egestas erat imperdiet sed euismod nisi porta lorem mollis. Eu feugiat pretium nibh ipsum consequat nisl vel pretium. Elit ut aliquam purus sit amet. Aliquet nibh praesent tristique magna sit. Dapibus ultrices in iaculis nunc.

Enim eu turpis egestas pretium aenean pharetra. Nunc sed blandit libero volutpat sed cras ornare arcu. Etiam erat velit scelerisque in. Purus semper eget duis at tellus at urna condimentum mattis. Sapien

Etiam erat velit scelerisque in. Purus semper eget duis at tellus at urna condimentum mattis. Sapien

## **Empty**

Insert a single-element container.

```
c = st.empty()  
st.write("This will show last")  
c.write("This will be replaced")  
c.write("This will show first")
```

Was this page helpful?

Yes  No

[edit](#)[Suggest edits](#)

[forum](#)

## **Still have questions?**

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: Media elements](#) [Next: st.sidebar](#) →

---

[Home](#)[Contact Us](#)[Community](#)





ImportError libGL.so.1 cannot open shared object file No such file or directory

# ImportError libGL.so.1 cannot open shared object file No such file or directory

## Problem

You receive the error `ImportError libGL.so.1 cannot open shared object file No such file or directory` when using OpenCV in your app deployed on Streamlit Cloud.

## Solution

If you use OpenCV in your app, include `opencv-python-headless` in your requirements file on Streamlit Cloud in place of `opencv_contrib_python` and `opencv-python`.

If `opencv-python` is a *required* (non-optional) dependency of your app or a dependency of a library used in your app, the above solution is not applicable. Instead, you can use the following solution:

Create a `packages.txt` file in your repo with the following line to install the apt-get dependency `libgl`:

`libgl1`

## Was this page helpful?

Yes

No

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: How to install a package not on PyPI or Conda but available on GitHub](#)

[Next: ModuleNotFoundError No module named](#) →

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.

## Documentation

*search*

Search

- *description*

### Streamlit library

- Get started  
*remove*
  - Main concepts
  - Installation
  - Create an app
- API reference  
*add*
- Advanced features  
*add*
- Components  
*add*
- Changelog
- Cheat sheet

- *cloud*

### Streamlit Cloud

- Get started  
*add*
- Trust and Security
- Release notes[open in new](#)
- Troubleshooting

- *school*

### Knowledge base

- Tutorials  
*add*
- Using Streamlit
- Streamlit Components
- Installing dependencies
- Deployment issues

- *Home/*
- *Streamlit library/*
- *Get started*

## **Get started**



This Get Started guide explains how Streamlit works, how to install Streamlit on your preferred operating system, and how to create your first Streamlit app!

### *description*

Main concepts introduces you to Streamlit's data model and development flow. You'll learn what makes Streamlit the most powerful way to build data apps, including the ability to display and style data, draw charts and maps, add interactive widgets, customize app layouts, cache computation, and define themes.

### *downloading*

[Installation](#) helps you set up your virtual environment and walks you through installing Streamlit on Windows, macOS, and Linux. Regardless of which package management tool and OS you're using, we recommend running the commands on this page in a virtual environment.

## [auto\\_awesome](#)

[Create an app](#) using Streamlit's core features to fetch and cache data, draw charts, plot information on a map, and use interactive widgets, like a slider, to filter results.

Was this page helpful?

Yes No

[edit](#)[Suggest edits](#)

[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

←[Previous: Streamlit library](#)[Next: Main concepts](#)→

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.



## Magic

8

Magic commands are a feature in Streamlit that allows you to write almost anything (markdown, data, charts) without having to type an explicit command at all. Just put the thing you want to show on its own line of code, and it will appear in your app. Here's an example:

```
# Draw a title and some text to the app:  
...  
# This is the document title  
  
This is some markdown.  
...  
  
import pandas as pd  
df = pd.DataFrame({'col1': [1,2,3]})  
df # 🤞 Draw the dataframe  
  
x = 10  
'x', x # 🤞 Draw the string 'x' and then the value of x  
  
# Also works with most supported chart types  
import matplotlib.pyplot as plt  
import numpy as np  
  
arr = np.random.normal(1, 1, size=100)  
fig, ax = plt.subplots()  
ax.hist(arr, bins=20)  
  
fig # 🤞 Draw a Matplotlib chart
```

Copy

## How Magic works

8

Any time Streamlit sees either a variable or literal value on its own line, it automatically writes that to your app using `st.write` (which you'll learn about later).

Also, magic is smart enough to ignore docstrings. That is, it ignores the strings at the top of files and functions.

If you prefer to call Streamlit commands more explicitly, you can always turn magic off in your

[`~/.streamlit/config.toml`](#) with the following setting:

```
[  
runner  
]  
magicEnabled = false  
Copy  
!
```

## Important

Right now, Magic only works in the main Python app file, not in imported files. See GitHub issue #288 for a discussion of the issues.

Was this page helpful?

 Yes  No

 Suggest edits



## Still have questions?

Our forums are full of helpful information and Streamlit experts.

 Previous: st.writeNext: Text elements  


---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# Main concepts

Working with Streamlit is simple. First you sprinkle a few Streamlit commands into a normal Python script, then you run it with `streamlit run` :

```
streamlit run your_script.py [-- script args]
```

As soon as you run the script as shown above, a local Streamlit server will spin up and your app will open in a new tab in your default web browser. The app is your canvas, where you'll draw charts, text, widgets, tables, and more.

What gets drawn in the app is up to you. For example `st.text` writes raw text to your app, and `st.line_chart` draws — you guessed it — a line chart. Refer to our [API documentation](#) to see all commands that are available to you.

## ★ Note

When passing your script some custom arguments, they must be passed after two dashes. Otherwise the arguments get interpreted as arguments to Streamlit itself.

Another way of running Streamlit is to run it as a Python module. This can be useful when configuring an IDE like PyCharm to work with Streamlit:

```
# Running
$ python -m streamlit run your_script.py

# is equivalent to:
$ streamlit run your_script.py
```

## ★ Tip

You can also pass a URL to `streamlit run`! This is great when combined with Github Gists. For example:

```
$ streamlit run https://raw.githubusercontent.com/streamlit/demo-uber-nyc-
```

## Development flow

Every time you want to update your app, save the source file. When you do that, Streamlit detects if there is a change and asks you whether you want to rerun your app. Choose "Always rerun" at the top-right of your screen to automatically update your app every time you change its source code.

This allows you to work in a fast interactive loop: you type some code, save it, try it out live, then type some more code, save it, try it out, and so on until you're happy with the results. This tight loop between coding and viewing results live is one of the ways Streamlit makes your life easier.

### ★ Tip

While developing a Streamlit app, it's recommended to lay out your editor and browser windows side by side, so the code and the app can be seen at the same time. Give it a try!

## Data flow

Streamlit's architecture allows you to write apps the same way you write plain Python scripts. To unlock this, Streamlit apps have a unique data flow: any time something must be updated on the screen, Streamlit reruns your entire Python script from top to bottom.

This can happen in two situations:

- Whenever you modify your app's source code.
- Whenever a user interacts with widgets in the app. For example, when dragging a slider, entering text in an input box, or clicking a button.

Whenever a callback is passed to a widget via the `on_change` (or `on_click`) parameter, the callback will always run before the rest of your script. For details on the Callbacks API, please refer to our [Session State API](#)

And to make all of this fast and seamless, Streamlit does some heavy lifting for you behind the scenes. A big player in this story is the `@st.cache` decorator, which allows developers to skip certain costly computations when their apps rerun. We'll cover caching later in this page.

## Display and style data

There are a few ways to display data (tables, arrays, data frames) in Streamlit apps. [Below](#), you will be introduced to `magic` and `st.write()`, which can be used to write anything from text to tables. After that, let's take a look at methods designed specifically for visualizing data.

## Use magic

You can also write to your app without calling any Streamlit methods. Streamlit supports "[magic commands](#)," which means you don't have to use `st.write()` at all! To see this in action try this snippet:

```
#####
# My first app
Here's our first attempt at using data to create a table:
#####

import streamlit as st
import pandas as pd
df = pd.DataFrame({
    'first column': [1, 2, 3, 4],
    'second column': [10, 20, 30, 40]
})

df
```

Any time that Streamlit sees a variable or a literal value on its own line, it automatically writes that to your app using `st.write()`. For more information, refer to the documentation on [magic commands](#).

## Write a data frame

Along with [magic commands](#), `st.write()` is Streamlit's "Swiss Army knife". You can pass almost anything to `st.write()`: text, data, Matplotlib figures, Altair charts, and more. Don't worry, Streamlit will figure it out and render things the right way.

```
import streamlit as st
import pandas as pd

st.write("Here's our first attempt at using data to create a table:")
st.write(pd.DataFrame({
    'first column': [1, 2, 3, 4],
    'second column': [10, 20, 30, 40]
}))
```

There are other data specific functions like `st.dataframe()` and `st.table()` that you can also use for displaying data. Let's understand when to use these features and how to add colors and styling to your data frames.

You might be asking yourself, "why wouldn't I always use `st.write()`?" There are a few reasons:

1. *Magic* and `st.write()` inspect the type of data that you've passed in, and then decide how to best render it in the app. Sometimes you want to draw it another way. For example, instead of drawing a dataframe as an interactive table, you may want to draw it as a static table by using `st.table(df)`.
2. The second reason is that other methods return an object that can be used and modified, either by adding data to it or replacing it.
3. Finally, if you use a more specific Streamlit method you can pass additional arguments to customize its behavior.

For example, let's create a data frame and change its formatting with a Pandas `Styler` object. In this example, you'll use Numpy to generate a random sample, and the `st.dataframe()` method to draw an interactive table.

### ❖ Note

This example uses Numpy to generate a random sample, but you can use Pandas DataFrames, Numpy arrays, or plain Python arrays.

```
import streamlit as st
import numpy as np

dataframe = np.random.randn(10, 20)
st.dataframe(dataframe)
```

Let's expand on the first example using the Pandas [Styler](#) object to highlight some elements in the interactive table.

```
import streamlit as st
import numpy as np
import pandas as pd

dataframe = pd.DataFrame(
    np.random.randn(10, 20),
    columns=('col %d' % i for i in range(20)))

st.dataframe(dataframe.style.highlight_max(axis=0))
```

Streamlit also has a method for static table generation: [st.table\(\)](#).

```
import streamlit as st
import numpy as np
import pandas as pd

dataframe = pd.DataFrame(
    np.random.randn(10, 20),
    columns=('col %d' % i for i in range(20)))
st.table(dataframe)
```

## Draw charts and maps

Streamlit supports several popular data charting libraries like [Matplotlib](#), [Altair](#), [deck.gl](#), and more. In this section, you'll add a bar chart, line chart, and a map to your app.

### Draw a line chart

You can easily add a line chart to your app with [st.line\\_chart\(\)](#). We'll generate a random sample using Numpy and then chart it.

```
import streamlit as st
import numpy as np
import pandas as pd
```

```
chart_data = pd.DataFrame(  
    np.random.randn(20, 3),  
    columns=['a', 'b', 'c'])  
  
st.line_chart(chart_data)
```

## Plot a map

With [st.map\(\)](#) you can display data points on a map. Let's use Numpy to generate some sample data and plot it on a map of San Francisco.

```
import streamlit as st  
import numpy as np  
import pandas as pd  
  
map_data = pd.DataFrame(  
    np.random.randn(1000, 2) / [50, 50] + [37.76, -122.4],  
    columns=['lat', 'lon'])  
  
st.map(map_data)
```

## Widgets

When you've got the data or model into the state that you want to explore, you can add in widgets like [st.slider\(\)](#), [st.button\(\)](#) or [st.selectbox\(\)](#). It's really straightforward — treat widgets as variables:

```
import streamlit as st  
x = st.slider('x') # 👈 this is a widget  
st.write(x, 'squared is', x * x)
```

On first run, the app above should output the text "0 squared is 0". Then every time a user interacts with a widget, Streamlit simply reruns your script from top to bottom, assigning the current state of the widget to your variable in the process.

For example, if the user moves the slider to position `10`, Streamlit will rerun the code above and set `x` to `10` accordingly. So now you should see the text "10 squared is 100".

Widgets can also be accessed by key, if you choose to specify a string to use as the unique key for the widget:

```
import streamlit as st
st.text_input("Your name", key="name")

# You can access the value at any point with:
st.session_state.name
```

Every widget with a key is automatically added to Session State. For more information about Session State, its association with widget state, and its limitations, see [Session State API Reference Guide](#).

## Use checkboxes to show/hide data

One use case for checkboxes is to hide or show a specific chart or section in an app. `st.checkbox()` takes a single argument, which is the widget label. In this sample, the checkbox is used to toggle a conditional statement.

```
import streamlit as st
import numpy as np
import pandas as pd

if st.checkbox('Show dataframe'):
    chart_data = pd.DataFrame(
        np.random.randn(20, 3),
        columns=['a', 'b', 'c'])

chart_data
```

## Use a selectbox for options

Use `st.selectbox` to choose from a series. You can write in the options you want, or pass through an array or data frame column.

Let's use the `df` data frame we created earlier.

```
import streamlit as st
import pandas as pd

df = pd.DataFrame({
    'first column': [1, 2, 3, 4],
    'second column': [10, 20, 30, 40]
})

option = st.selectbox(
    'Which number do you like best?',
    df['first column'])

>You selected: ', option
```

## Layout

Streamlit makes it easy to organize your widgets in a left panel sidebar with `st.sidebar`. Each element that's passed to `st.sidebar` is pinned to the left, allowing users to focus on the content in your app while still having access to UI controls.

For example, if you want to add a selectbox and a slider to a sidebar, use `st.sidebar.slider` and `st.sidebar.selectbox` instead of `st.slider` and `st.selectbox`:

```
import streamlit as st

# Add a selectbox to the sidebar:
add_selectbox = st.sidebar.selectbox(
    'How would you like to be contacted?',
    ('Email', 'Home phone', 'Mobile phone')
)

# Add a slider to the sidebar:
add_slider = st.sidebar.slider(
    'Select a range of values',
    0.0, 100.0, (25.0, 75.0)
)
```

Beyond the sidebar, Streamlit offers several other ways to control the layout of your app. `st.columns` lets you place widgets side-by-side, and `st.expander` lets you conserve space by hiding away large content.

```
import streamlit as st

left_column, right_column = st.columns(2)
# You can use a column just like st.sidebar:
left_column.button('Press me!')

# Or even better, call Streamlit functions inside a "with" block:
with right_column:
    chosen = st.radio(
        'Sorting hat',
        ("Gryffindor", "Ravenclaw", "Hufflepuff", "Slytherin"))
    st.write(f"You are in {chosen} house!")
```

## ★ Note

`st.echo` and `st.spinner` are not currently supported inside the sidebar or layout options. Rest assured, though, we're currently working on adding support for those too!

## Show progress

When adding long running computations to an app, you can use `st.progress()` to display status in real time.

First, let's import time. We're going to use the `time.sleep()` method to simulate a long running computation:

```
import time
```

Now, let's create a progress bar:

```
import streamlit as st
import time

'Starting a long computation...'
```

```
# Add a placeholder
latest_iteration = st.empty()
bar = st.progress(0)

for i in range(100):
    # Update the progress bar with each iteration.
    latest_iteration.text(f'Iteration {i+1}')
    bar.progress(i + 1)
    time.sleep(0.1)

'...and now we're done!'
```

## Themes

Streamlit supports Light and Dark themes out of the box. Streamlit will first check if the user viewing an app has a Light or Dark mode preference set by their operating system and browser. If so, then that preference will be used. Otherwise, the Light theme is applied by default.

You can also change the active theme from "≡" → "Settings".

The screenshot shows the Streamlit welcome page with the Light theme selected. On the left, there's a sidebar with a dropdown menu labeled "Choose a demo" and a green button that says "Select a demo above.". The main content area has a title "Welcome to Streamlit!" with a yellow clapping hands emoji. Below it, a paragraph of text and a tip about selecting a demo from the dropdown. There are sections for "Want to learn more?" with a list of links, and "See more complex demos" with another list of links. The overall interface is clean and modern.

Want to add your own theme to an app? The "Settings" menu has a theme editor accessible by clicking on "Edit active theme". You can use this editor to try out different colors and see your app update live.

The screenshot shows the Streamlit theme editor interface. On the left, there is a dropdown menu labeled "Choose a demo" with a placeholder "-". Below it is a green button with the text "Select a demo above.". In the center, there is a large heading "Welcome to Streamlit!" with a yellow clapping hands emoji. Below the heading, a text block says "Streamlit is an open-source app framework built specifically for Machine Learning and Data Science projects." A tip message follows: "👉 Select a demo from the dropdown on the left to see some examples of what Streamlit can do!". Underneath, there is a section titled "Want to learn more?" with three bullet points: "Check out [streamlit.io](#)", "Jump into our [documentation](#)", and "Ask a question in our [community forums](#)". Another section titled "See more complex demos" contains two bullet points: "Use a neural net to [analyze the Udacity Self-driving Car Image Dataset](#)" and "Explore a [New York City rideshare dataset](#)". On the right side, there is a sidebar with a red header bar containing the Streamlit logo and a menu icon. The sidebar lists several options: "Rerun" (with a "R" icon), "Clear cache" (with a "C" icon), "Deploy this app", "Record a screencast", "Documentation", "Ask a question", "Report a bug", "Streamlit for Teams", "Settings", and "About".

When you're happy with your work, themes can be saved by setting config options in the `[theme]` config section. After you've defined a theme for your app, it will appear as "Custom Theme" in the theme selector and will be applied by default instead of the included Light and Dark themes.

More information about the options available when defining a theme can be found in the [theme option documentation](#).

## ★ Note

The theme editor menu is available only in local development. If you've deployed your app using Streamlit Cloud, the "Edit active theme" button will no longer be displayed in the "Settings" menu.

## ★ Tip

Another way to experiment with different theme colors is to turn on the "Run on save" option, edit your config.toml file, and watch as your app reruns with the new theme colors applied.

## ! Important

We're developing new cache primitives that are easier to use and much faster than `@st.cache` .  To learn more, read [Experimental cache primitives](#).

The Streamlit cache allows your app to execute quickly even when loading data from the web, manipulating large datasets, or performing expensive computations.

To use the cache, wrap functions with the `@st.cache` decorator:

```
import streamlit as st

@st.cache # 👈 This function will be cached
def my_slow_function(arg1, arg2):
    # Do something really slow in here!
    return the_output
```

When you mark a function with the `@st.cache` decorator, it tells Streamlit that whenever the function is called it needs to check a few things:

1. The input parameters that you called the function with
2. The value of any external variable used in the function
3. The body of the function
4. The body of any function used inside the cached function

If this is the first time Streamlit has seen these four components with these exact values and in this exact combination and order, it runs the function and stores the result in a local cache. Then, next time the cached function is called, if none of these components changed, Streamlit will skip executing the function altogether and, instead, return the output previously stored in the cache.

For more information about the Streamlit cache, its configuration parameters, and its limitations, see [Caching](#).

Now that you know a little more about all the individual pieces, let's close the loop and review how it works together:

1. Streamlit apps are Python scripts that run from top to bottom
2. Every time a user opens a browser tab pointing to your app, the script is re-executed
3. As the script executes, Streamlit draws its output live in a browser
4. Scripts use the Streamlit cache to avoid recomputing expensive functions, so updates happen very fast
5. Every time a user interacts with a widget, your script is re-executed and the output value of that widget is set to the new value during that run.



## Was this page helpful?

Yes

No

Suggest edits



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

[← Previous: Get started](#)

[Next: Installation →](#)

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# Media elements

It's easy to embed images, videos, and audio files directly into your Streamlit apps.



## Image

Display an image or list of images.

```
st.image(numpy_array)  
st.image(image_bytes)  
st.image(file)  
st.image("https://example.com/myimage.jpg")
```



## Audio

Display an audio player.

```
st.audio(numpy_array)  
st.audio(audio_bytes)  
st.audio(file)  
st.audio("https://example.com/myaudio.mp3", format="audio/mp3")
```

## Video

Display a video player.

```
st.video(numpy_array)
st.video(video_bytes)
st.video(file)
st.video("https://example.com/myvideo.mp4", format="video/mp4")
```

## Was this page helpful?

 Yes     No

 [Suggest edits](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

[← Previous: Input widgets](#)

[Next: st.image →](#)



Copyright © 2022, Streamlit Inc.



# ModuleNotFoundError: No module named

## Problem

You receive the error `ModuleNotFoundError: No module named` when you deploy an app on [Streamlit Cloud](#).

## Solution

This error occurs when you import a module on Streamlit Cloud that isn't included in your requirements file. Any external [Python dependencies](#) that are not distributed with a [standard Python installation](#) should be included in your requirements file.

E.g. You will see `ModuleNotFoundError: No module named 'sklearn'` if you don't include `scikit-learn` in your requirements file and `import sklearn` in your app.

Related forum posts:

- <https://discuss.streamlit.io/t/getting-error-modulenotfounderror-no-module-named-beautifulsoup/9126>
- <https://discuss.streamlit.io/t/modulenotfounderror-no-module-named-vega-datasets/16354>

## Was this page helpful?

Yes

No

[Suggest edits](#)



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

← **Previous:** ImportError libGL.so.1 cannot open shared object file No such file or directory

**Next:** ERROR No matching distribution found for →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# Connect Streamlit to MongoDB

## Introduction

This guide explains how to securely access a MongoDB database from Streamlit Cloud. It uses the [PyMongo](#) library and Streamlit's [secrets management](#).

## Create a MongoDB Database

### ★ Note

If you already have a database that you want to use, feel free to [skip to the next step](#).

First, follow the official tutorials to [install MongoDB](#), [set up authentication](#) (note down the username and password!), and [connect to the MongoDB instance](#). Once you are connected, open the `mongo` shell and enter the following two commands to create a collection with some example values:

```
use mydb
db.mycollection.insertMany( [{"name" : "Mary", "pet": "dog"}, {"name" : "John", "pet": "cat"}])
```

## Add username and password to your local app secrets

Your local Streamlit app will read secrets from a file `.streamlit/secrets.toml` in your app's root directory. Create this file if it doesn't exist yet and add the database information as shown below:

```
# .streamlit/secrets.toml
```

```
[mongo]
host = "localhost"
port = 27017
username = "xxx"
password = "xxx"
```

## ! Important

Add this file to [.gitignore](#) and don't commit it to your Github repo!

## Copy your app secrets to the cloud

As the [secrets.toml](#) file above is not committed to Github, you need to pass its content to your deployed app (on Streamlit Cloud) separately. Go to the [app dashboard](#) and in the app's dropdown menu, click on **Edit Secrets**. Copy the content of [secrets.toml](#) into the text area. More information is available at [Secrets Management](#).

The screenshot shows a Streamlit app settings page titled "awesomestreamlit's apps". On the left, there's a sidebar with "App settings", "Sharing", and "Secrets" options. A modal window titled "Secrets" is open, explaining how to provide environment variables and other secrets using TOML format. It contains sample TOML code:

```
DB_USERNAME = "myuser"  
DB_TOKEN = "abcdef"  
  
[some_section]  
some_key = 1234
```

A "Save" button is at the bottom right of the modal.

## Add PyMongo to your requirements file

Add the PyMongo package to your `requirements.txt` file, preferably pinning its version (replace `x.x.x` with the version you want installed):

```
# requirements.txt  
pymongo==x.x.x
```

## Write your Streamlit app

Copy the code below to your Streamlit app and run it. Make sure to adapt the name of your database and collection.

```
# streamlit_app.py

import streamlit as st
import pymongo

# Initialize connection.
# Uses st.experimental_singleton to only run once.
@st.experimental_singleton
def init_connection():
    return pymongo.MongoClient(**st.secrets["mongo"])

client = init_connection()

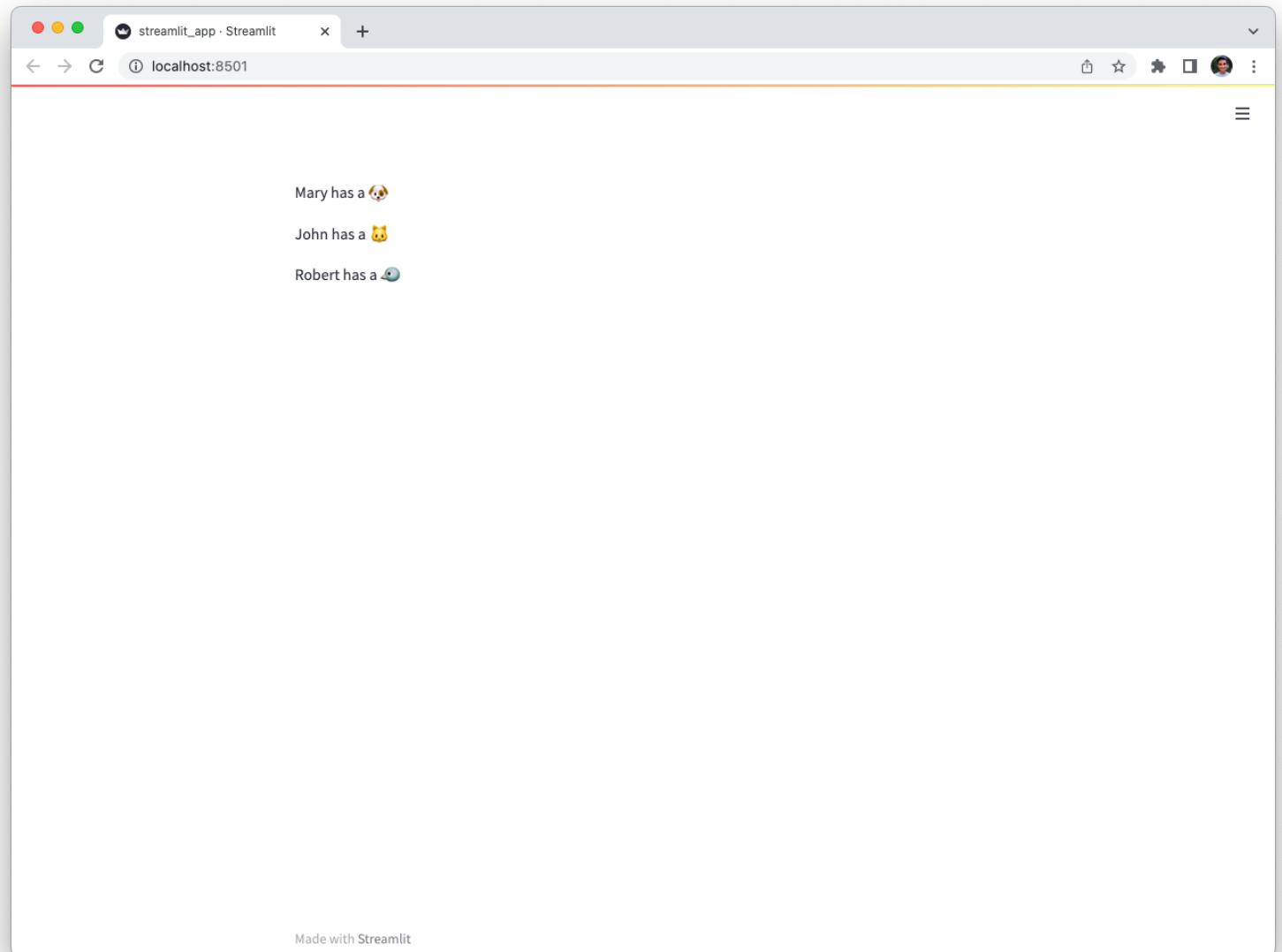
# Pull data from the collection.
# Uses st.experimental_memo to only rerun when the query changes or after 10 minutes.
@st.experimental_memo(ttl=600)
def get_data():
    db = client.mydb
    items = db.mycollection.find()
    items = list(items) # make hashable for st.experimental_memo
    return items

items = get_data()

# Print results.
for item in items:
    st.write(f"{item['name']} has a :{item['pet']}:")
```

See `st.experimental_memo` above? Without it, Streamlit would run the query every time the app reruns (e.g. on a widget interaction). With `st.experimental_memo`, it only runs when the query changes or after 10 minutes (that's what `ttl` is for). Watch out: If your database updates more frequently, you should adapt `ttl` or remove caching so viewers always see the latest data. Read more about caching [here](#).

If everything worked out (and you used the example data we created above), your app should look like this:



## Was this page helpful?

 Yes  No

 [Suggest edits](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# Mutate charts

Sometimes you display a chart or dataframe and want to modify live as the app runs (for example, in a loop).

Depending on what you're looking for, there are 3 different ways to do this:

1. Using `st.empty` to replace a single element.
2. Using `st.container` or `st.columns` to replace multiple elements.
3. Using `add_rows` to append data to specific types of elements.

Here we discuss that last case.

## `st.add_rows`

**v1.9.0**

Concatenate a dataframe to the bottom of the current one.

### Function signature

```
element.add_rows(self, data=None, **kwargs)
```

### Parameters

`data (pandas.DataFrame, pandas.Styler, pyarrow.Table, numpy.ndarray, Iterable, dict, or None)`

Table to concat. Optional. Pyarrow tables are not supported by Streamlit's legacy DataFrame serialization (i.e. with `config.dataFrameSerialization = "legacy"`). To use pyarrow tables, please enable pyarrow by changing the config setting, `config.dataFrameSerialization = "arrow"`.

`**kwargs (pandas.DataFrame, numpy.ndarray, Iterable, dict, or None)`

The named dataset to concat. Optional. You can only pass in 1 dataset (including the one in the data parameter).

## Example

```
df1 = pd.DataFrame(  
    np.random.randn(50, 20),  
    columns=(('col %d' % i) for i in range(20)))  
  
my_table = st.table(df1)  
  
df2 = pd.DataFrame(  
    np.random.randn(50, 20),  
    columns=(('col %d' % i) for i in range(20)))  
  
my_table.add_rows(df2)  
# Now the table shown in the Streamlit app contains the data for  
# df1 followed by the data for df2.
```

You can do the same thing with plots. For example, if you want to add more data to a line chart:

```
# Assuming df1 and df2 from the example above still exist...  
my_chart = st.line_chart(df1)  
my_chart.add_rows(df2)  
# Now the chart shown in the Streamlit app contains the data for  
# df1 followed by the data for df2.
```

And for plots whose datasets are named, you can pass the data with a keyword argument where the key is the name:

```
my_chart = st.vega_lite_chart({  
    'mark': 'line',  
    'encoding': {'x': 'a', 'y': 'b'},  
    'datasets': {  
        'some_fancy_name': df1, # <-- named dataset  
    },  
    'data': {'name': 'some_fancy_name'},  
},  
my_chart.add_rows(some_fancy_name=df2) # <-- name used as keyword
```

## Was this page helpful?

 Yes

 No

 [Suggest edits](#)

---

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[← Previous: Utilities](#)

[Next: State management →](#)

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# Connect Streamlit to MySQL

## Introduction

This guide explains how to securely access a MySQL database from Streamlit Cloud. It uses the [mysql-connector-python](#) library and Streamlit's [secrets management](#).

## Create a MySQL database

### ★ Note

If you already have a database that you want to use, feel free to [skip to the next step](#).

First, follow [this tutorial](#) to install MySQL and start the MySQL server (note down the username and password!). Once your MySQL server is up and running, connect to it with the `mysql` client and enter the following commands to create a database and a table with some example values:

```
CREATE DATABASE pets;
```

```
USE pets;
```

```
CREATE TABLE mytable (
    name      varchar(80),
    pet       varchar(80)
);
```

```
INSERT INTO mytable VALUES ('Mary', 'dog'), ('John', 'cat'), ('Robert', 'bird')
```

## Add username and password to your local app secrets

Your local Streamlit app will read secrets from a file `.streamlit/secrets.toml` in your app's root directory. Create this file if it doesn't exist yet and add the database name, user, and password of your MySQL server as shown below:

```
# .streamlit/secrets.toml
```

```
[mysql]
host = "localhost"
port = 3306
database = "xxx"
user = "xxx"
password = "xxx"
```

### ! Important

Add this file to `.gitignore` and don't commit it to your Github repo!

## Copy your app secrets to the cloud

As the `secrets.toml` file above is not committed to Github, you need to pass its content to your deployed app (on Streamlit Cloud) separately. Go to the [app dashboard](#) and in the app's dropdown menu, click on **Edit Secrets**. Copy the content of `secrets.toml` into the text area. More information is available at [Secrets Management](#).

The screenshot shows a Streamlit app titled "awesomestreamlit's apps" on share.streamlit.io. The left sidebar has options for "App settings", "Sharing", and "Secrets". A modal window titled "Secrets" is open, explaining how to provide environment variables and other secrets using TOML format. It contains sample code:

```
DB_USERNAME = "myuser"  
DB_TOKEN = "abcdef"  
  
[some_section]  
some_key = 1234
```

A "Save" button is at the bottom right of the modal.

## Add mysql-connector-python to your requirements file

Add the [mysql-connector-python](#) package to your `requirements.txt` file, preferably pinning its version (replace `x.x.x` with the version you want installed):

```
# requirements.txt  
mysql-connector-python==x.x.x
```

## Write your Streamlit app

Copy the code below to your Streamlit app and run it. Make sure to adapt `query` to use the name of your table.

```

# streamlit_app.py

import streamlit as st
import mysql.connector

# Initialize connection.
# Uses st.experimental_singleton to only run once.
@st.experimental_singleton
def init_connection():
    return mysql.connector.connect(**st.secrets["mysql"])

conn = init_connection()

# Perform query.
# Uses st.experimental_memo to only rerun when the query changes or after 10 minutes.
@st.experimental_memo(ttl=600)
def run_query(query):
    with conn.cursor() as cur:
        cur.execute(query)
    return cur.fetchall()

rows = run_query("SELECT * from mytable;")

# Print results.
for row in rows:
    st.write(f"{row[0]} has a :{row[1]}:")

```

See `st.experimental_memo` above? Without it, Streamlit would run the query every time the app reruns (e.g. on a widget interaction). With `st.experimental_memo`, it only runs when the query changes or after 10 minutes (that's what `ttl` is for). Watch out: If your database updates more frequently, you should adapt `ttl` or remove caching so viewers always see the latest data. Read more about caching [here](#).

If everything worked out (and you used the example table we created above), your app should look like this:

Mary has a 🐄

John has a 🐱

Robert has a 🐈

Made with Streamlit

## Was this page helpful?

👍 Yes

👎 No

[Suggest edits](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# ERROR: No matching distribution found for

## Problem

You receive the error **ERROR: No matching distribution found for** when you deploy an app on Streamlit Cloud.

## Solution

This error occurs when you deploy an app on Streamlit Cloud and have one or more of the following issues with your Python dependencies in your requirements file:

1. The package is part of the Python Standard Library. E.g. You will see **ERROR: No matching distribution found for base64** if you include base64 in your requirements file, as it is part of the Python Standard Library. The solution is to not include the package in your requirements file. Only include packages in your requirements file that are not distributed with a standard Python installation.
2. The package name in your requirements file is misspelled. Double-check the package name before including it in your requirements file.
3. The package does not support the operating system on which your Streamlit app is running. E.g. You see **ERROR: No matching distribution found for pywin32** while deploying to Streamlit Cloud. The pywin32 module provides access to many of the Windows APIs from Python. Apps deployed to Streamlit Cloud are executed in a Linux environment. As such, pywin32 fails to install on non-Windows systems, including on Streamlit Cloud. The solution is to either exclude pywin32 from your requirements file, or deploy your app on a cloud service offering Windows machines.

Related forum posts:

- <https://discuss.streamlit.io/t/error-no-matching-distribution-found-for-base64/15758>

- <https://discuss.streamlit.io/t/error-could-not-find-a-version-that-satisfies-the-requirement-pywin32-301-from-versions-none/15343/2>

**Was this page helpful?**

 Yes

 No

 [Suggest edits](#)

---

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

← **Previous:** [ModuleNotFoundError No module named](#)

**Next:** [Deployment issues](#) →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



Home / Knowledge base / Streamlit Components /

What types of things aren't possible with Streamlit Components?

# What types of things aren't possible with Streamlit Components?

Because each Streamlit Component gets mounted into its own sandboxed iframe, this implies a few limitations on what is possible with Components:

- **Can't communicate with other Components:** Components can't contain (or otherwise communicate with) other components, so Components cannot be used to build something like `grid_layout`
- **Can't modify CSS:** A Component can't modify the CSS that the rest of the Streamlit app uses, so you can't create something like `dark_mode`
- **Can't add/remove elements:** A Component can't add or remove other elements of a Streamlit app, so you couldn't make something like `remove_streamlit_hamburger_menu`

Was this page helpful?

Yes    No

[Suggest edits](#)



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

**Next:** Installing dependencies →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.

## Documentation

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [\*cloud\*](#)

### [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notes open in new](#)
- [Troubleshooting](#)

- [\*school\*](#)

### [Knowledge base](#)

- [Tutorials](#)  
*add*
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [\*Home\*](#)/
- [\*Knowledge base\*](#)/
- [\*Using Streamlit\*](#)/
- [\*What is the path of Streamlit's config.toml file?\*](#)

## **What is the path of Streamlit's config.toml file?**

8

A global config file is found at `~/.streamlit/config.toml` for macOS/Linux or `%userprofile%/.streamlit/config.toml` for Windows.

A per-project config file can be created at `$CWD/.streamlit/config.toml`, where `$CWD` is the folder you're running Streamlit from.

Click [here](#) to learn more about Streamlit configuration options.

Was this page helpful?

Yes  No

[edit](#) [Suggest edits](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: How to insert elements out of order?](#) [Next: How can I make st.pydeck\\_chart use custom Mapbox styles?](#) →

---

[Home](#) [Contact Us](#) [Community](#)



Copyright © 2022, Streamlit Inc.



# Optimize performance

## Caching

Function decorator to memoize function executions.

```
@st.cache(ttl=3600)  
def run_long_computation(arg1, arg2):  
    # Do stuff here  
    return computation_output
```

## Memo

Experimental function decorator to memoize function executions.

```
@st.experimental_memo  
def fetch_and_clean_data(url):  
    # Fetch data from URL here, and then clean it up.  
    return data
```

## Singleton

Experimental function decorator to store singleton objects.

```
@st.experimental_singleton  
def get_database_session(url):  
    # Create a database session object that points to the URL.  
    return session
```

Was this page helpful?

Yes

No

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: State management](#)

[Next: st.cache](#) →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [\*cloud\*](#)

### [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notes](#)*open in new*
- [Troubleshooting](#)

- [\*school\*](#)

### [Knowledge base](#)

- [Tutorials](#)  
*remove*
  - [Connect to data sources](#)  
*remove*
    - [AWS S3](#)
    - [BigQuery](#)
    - [Snowflake](#)
    - [Microsoft SQL Server](#)
    - [Firestore](#)*open in new*
    - [MongoDB](#)
    - [MySQL](#)
    - [PostgreSQL](#)
    - [Tableau](#)
    - [Private Google Sheet](#)
    - [Public Google Sheet](#)
    - [TigerGraph](#)
    - [Deta Base](#)
    - [Supabase](#)
    - [Google Cloud Storage](#)
  - [Session State basics](#)
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [\*Home\*](#)/
- [\*Knowledge base\*](#)/
- [\*Tutorials\*](#)/
- [\*Connect to data sources\*](#)/
- [\*PostgreSQL\*](#)

## [Connect Streamlit to PostgreSQL](#)

*8*

## **Introduction**

*8*

This guide explains how to securely access a PostgreSQL database from Streamlit Cloud. It uses the [psycopg2](#) library and Streamlit's [secrets management](#).

## **Create a PostgreSQL database**

*8*

*push\_pin*

### **Note**

If you already have a database that you want to use, feel free to [skip to the next step](#).

First, follow [this tutorial](#) to install PostgreSQL and create a database (note down the database name, username, and password!). Open the SQL Shell (`psql`) and enter the following two commands to create a table with some example values:

```
CREATE TABLE mytable (
    name        varchar(80),
    pet         varchar(80)
);
```

```
INSERT INTO mytable VALUES ('Mary', 'dog'), ('John', 'cat'), ('Robert', 'bird');
```

## Add username and password to your local app secrets

8

Your local Streamlit app will read secrets from a file `.streamlit/secrets.toml` in your app's root directory. Create this file if it doesn't exist yet and add the name, user, and password of your database as shown below:

```
# .streamlit/secrets.toml
```

```
[postgres]
host = "localhost"
port = 5432
dbname = "xxx"
user = "xxx"
password = "xxx"
```

*priority\_high*

### Important

Add this file to `.gitignore` and don't commit it to your Github repo!

## Copy your app secrets to the cloud

8

As the `secrets.toml` file above is not committed to Github, you need to pass its content to your deployed app (on Streamlit Cloud) separately. Go to the [app dashboard](#) and in the app's dropdown menu, click on **Edit Secrets**. Copy the content of `secrets.toml` into the text area. More information is available at [Secrets Management](#).

The screenshot shows the Streamlit Cloud interface. At the top, there's a browser-like header with tabs for 'awesomestreamlit's apps · Streamlit' and a search bar containing 'share.streamlit.io'. On the right, there are icons for settings, a profile, and a red notification badge. Below the header, the main title is 'awesomestreamlit's apps'. On the left, a sidebar has three items: 'App settings', 'Sharing', and 'Secrets', with 'Secrets' currently selected. A modal window titled 'Secrets' is open in the center. It contains instructions: 'Provide environment variables and other secrets to your app using TOML format. This information is encrypted and served securely to your app at runtime. Learn more about Secrets in our [docs](#). Changes take around a minute to propagate.' Below the instructions is a code editor with the following TOML content:

```
DB_USERNAME = "myuser"
DB_TOKEN = "abcdef"

[some_section]
some_key = 1234
```

At the bottom right of the modal is a blue 'Save' button.

## Add psycopg2 to your requirements file

8

Add the [psycopg2](#) package to your `requirements.txt` file, preferably pinning its version (replace `xxxx` with the version you want installed):

```
# requirements.txt  
psycopg2-binary==x.x.x
```

## Write your Streamlit app

8

Copy the code below to your Streamlit app and run it. Make sure to adapt `query` to use the name of your table.

```
# streamlit_app.py  
  
import streamlit as st  
import psycopg2  
  
# Initialize connection.  
# Uses st.experimental_singleton to only run once.  
@st.experimental_singleton  
def init_connection():  
    return psycopg2.connect(**st.secrets["postgres"])  
  
conn = init_connection()  
  
# Perform query.  
# Uses st.experimental_memo to only rerun when the query changes or after 10 min.  
@st.experimental_memo(ttl=600)  
def run_query(query):  
    with conn.cursor() as cur:  
        cur.execute(query)  
        return cur.fetchall()  
  
rows = run_query("SELECT * from mytable;")  
  
# Print results.  
for row in rows:  
    st.write(f"{row[0]} has a :{row[1]}:")
```

See `st.experimental_memo` above? Without it, Streamlit would run the query every time the app reruns (e.g. on a widget interaction). With `st.experimental_memo`, it only runs when the query changes or after 10 minutes (that's what `ttl` is for). Watch out: If your database updates more frequently, you should adapt `ttl` or remove caching so viewers always see the latest data. Read more about caching [here](#).

If everything worked out (and you used the example table we created above), your app should look like this:

Mary has a 🐶

John has a 🐱

Robert has a 🐈

Made with Streamlit

Was this page helpful?

[thumb\\_upYes](#)  [thumb\\_downNo](#)  
[edit](#)[Suggest edits](#)  
[forum](#)

### Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

←[Previous: MySQL](#)[Next: Tableau](#)→

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.

# Documentation



Search

- 

Streamlit library

- Get started



- API reference



- Advanced features



- Theming
- Configuration
- Optimize performance with st.cache
- Experimental cache primitives
- Add statefulness to apps
- Widget semantics
- Pre-release features
- Working with timezones

- Components



- Changelog

- Cheat sheet

- 

Streamlit Cloud

- Get started



- Trust and Security



- Release notes

- Troubleshooting



Knowledge base

- Tutorials

+

- Using Streamlit

- Streamlit Components

- Installing dependencies

- Deployment issues

[Home](#) / [Streamlit library](#) / [Advanced features](#) / **Pre release features**

# Pre-release features

At Streamlit, we like to move quick while keeping things stable. In our latest effort to move even faster without sacrificing stability, we're offering our bold and fearless users two ways to try out Streamlit's bleeding-edge features:

1. Nightly releases
2. Beta and experimental features

## Nightly releases

At the end of each day (at night 🌙), our bots run automated tests against the latest Streamlit code and, if everything looks good, it publishes them as the `streamlit-nightly` package. This means the nightly build includes all our latest features, bug fixes, and other enhancements on the same day they land on our codebase.

### How does this differ from official releases?

Official Streamlit releases go not only through both automated tests but also rigorous manual testing, while nightly releases only have automated tests. It's important to keep in mind that new features introduced in nightly releases often lack polish. In our official releases, we always make double-sure all new features are ready for prime time.

### How do I use the nightly release?

All you need to do is install the `streamlit-nightly` package:

```
pip uninstall streamlit  
pip install streamlit-nightly --upgrade
```

## ! Warning

You should never have both `streamlit` and `streamlit-nightly` installed in the same environment!

### Why should I use the nightly release?

Because you can't wait for official releases, and you want to help us find bugs early!

### Why shouldn't I use the nightly release?

While our automated tests have high coverage, there's still a significant likelihood that there will be some bugs in the nightly code.

### Can I choose which nightly release I want to install?

If you'd like to use a specific version, you can find the version number in our Release history. Specify the desired version using `pip` as usual: `pip install streamlit-nightly==x.yy.zz-123456`.

### Can I compare changes between releases?

If you'd like to review the changes for a nightly release, you can use the comparison tool on GitHub.

## Beta and Experimental Features

In addition to nightly releases, we also have two naming conventions for less stable Streamlit features: `st.beta_` and `st.experimental_`. These distinctions are prefixes we attach to our function names to make sure their status is clear to everyone.

Here's a quick rundown of what you get from each naming convention:

- **st:** this is where our core features like `st.write` and `st.dataframe` live. If we ever make backward-incompatible changes to these, they will take place gradually and with months of announcements and warnings.
- **beta:** this is where all new features land before they become part of Streamlit core. This gives you a chance to try the next big thing we're cooking up weeks or months before we're ready to stabilize its API.

- **experimental**: this is where we'll put features that may or may not ever make it into Streamlit core. We don't know whether these features have a future, but we want you to have access to everything we're trying, and work with us to figure them out.

The main difference between `beta_` and `experimental_` is that beta features are expected to make it into Streamlit core at some point soon, while experimental features may never make it.

## Beta

Features with the `beta_` naming convention are all scheduled to become part of Streamlit core. While in beta, a feature's API and behaviors may not be stable, and it's possible they could change in ways that aren't backward-compatible.

### The lifecycle of a beta feature

1. A feature is added with the `beta_` prefix.
2. The feature's API stabilizes and the feature is *cloned* without the `beta_` prefix, so it exists as both `st` and `beta_`. At this point, users will see a warning when using the version of the feature with the `beta_` prefix -- but the feature will still work.
3. At some point, the code of the `beta_`-prefixed feature is *removed*, but there will still be a stub of the function prefixed with `beta_` that shows an error with appropriate instructions.
4. Finally, at a later date the `beta_` version is removed.

## Experimental

Features with the `experimental_` naming convention are things that we're still working on or trying to understand. If these features are successful, at some point they'll become part of Streamlit core, by moving to the `beta_` naming convention and then to Streamlit core. If unsuccessful, these features are removed without much notice.

### ! Warning

Experimental features and their APIs may change or be removed at any time.

### The lifecycle of an experimental feature

1. A feature is added with the `experimental_` prefix.
2. The feature is potentially tweaked over time, with possible API/behavior breakages.

3. At some point, we either promote the feature to `beta_` or remove it from `experimental_`. Either way, we leave a stub in `experimental_` that shows an error with instructions.

## Was this page helpful?

 Yes     No

 [Suggest edits](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

[← Previous: Widget semantics](#)

[Next: Working with timezones →](#)

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [\*cloud\*](#)

### [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notes](#)*open in new*
- [Troubleshooting](#)

- [\*school\*](#)

### [Knowledge base](#)

- [Tutorials](#)  
*remove*
  - [Connect to data sources](#)  
*remove*
    - [AWS S3](#)
    - [BigQuery](#)
    - [Snowflake](#)
    - [Microsoft SQL Server](#)
    - [Firestore](#)*open in new*
    - [MongoDB](#)
    - [MySQL](#)
    - [PostgreSQL](#)
    - [Tableau](#)
    - [Private Google Sheet](#)
    - [Public Google Sheet](#)
    - [TigerGraph](#)
    - [Deta Base](#)
    - [Supabase](#)
    - [Google Cloud Storage](#)
  - [Session State basics](#)
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home/](#)
- [Knowledge base/](#)
- [Tutorials/](#)
- [Connect to data sources/](#)
- [Private Google Sheet](#)

## Connect Streamlit to a private Google Sheet

*8*

## Introduction

*8*

This guide explains how to securely access a private Google Sheet from Streamlit Cloud. It uses the [gsheetsdb](#) library and Streamlit's [secrets management](#).

If you are fine with enabling link sharing for your Google Sheet (i.e. everyone with the link can view it), the guide [Connect Streamlit to a public Google Sheet](#) shows a simpler method of doing this. If your Sheet contains sensitive information and you cannot enable link sharing, keep on reading.

## Create a Google Sheet

*8*

*push\_pin*

### Note

If you already have a database that you want to use, feel free to [skip to the next step](#).

The screenshot shows a Google Sheets document with the title "streamlit-gsheets-demo-private". The spreadsheet has a single sheet named "Sheet1". The data is organized into two columns: "name" and "pet". The first four rows contain data: Mary (dog), John (cat), and Robert (bird). Row 5 is currently selected, indicated by a blue border around the entire row. The column headers are A, B, C, D, E, and F. The row numbers on the left range from 1 to 16. The top menu bar includes File, Edit, View, Insert, Format, Data, Tools, Add-ons, Help, and Share. On the right side of the interface, there are several icons for different Google services like Google Sheets, Google Slides, Google Forms, and Google Sheets Help.

	A	B	C	D	E	F
1	<b>name</b>	<b>pet</b>				
2	Mary	dog				
3	John	cat				
4	Robert	bird				
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						

## Enable the Sheets API

8

Programmatic access to Google Sheets is controlled through [Google Cloud Platform](#). Create an account or sign in and head over to the [APIs & Services dashboard](#) (select or create a project if asked). As shown below, search for the Sheets API and enable it:

The screenshot shows the Google Cloud Platform API & Services page for the project "My First Project". The top navigation bar includes a search icon, a gift icon, a question mark icon, a green circular icon with the number 2, and a user profile icon. A red circle highlights the "+ ENABLE APIs AND SERVICES" button, which is located above the main content area. The main content area displays traffic data from April 11 to May 02, showing rates of 0.04/s, 0.03/s, 0.02/s, and 0.01/s. Below the traffic chart is an "Errors" section with a 100% completion rate.

The screenshot shows the Google Cloud Platform API Library interface. The browser title bar reads "API API Library - My First Project". The address bar shows "console.cloud.google.com". The main header has a blue bar with the "Google Cloud Platform" logo, "My First Project" dropdown, and various navigation icons. Below the header, there's a search bar with "Search" and "Sheets" input fields, and a close button. On the left, a sidebar titled "Filter by" lists categories: "CATEGORY", "CRM (1)", and "Google Workspace (1)". The main content area displays "1 result" for the "Google Sheets API". It features a green icon of a document with a grid, the text "Google Sheets API" and "Google", and a brief description: "The Sheets API gives you full control over the content and appearance of your".

The screenshot shows a browser window with the URL `console.cloud.google.com`. The title bar says "API Google Sheets API – APIs & Services". The main content area is titled "Google Sheets API" and is described as a "Google" service. It states: "The Sheets API gives you full control over the content and appearance of your spreadsheet data." Below this are two buttons: "ENABLE" (which is circled in red) and "TRY THIS API". At the bottom of the page are three tabs: "OVERVIEW" (which is underlined), "DOCUMENTATION", and "SUPPORT".

## Overview

Reads and writes Google Sheets.

## Create a service account & key file

8

To use the Sheets API from Streamlit Cloud, you need a Google Cloud Platform service account (a special account type for programmatic data access). Go to the [Service Accounts page](#) and create an account with the **Viewer** permission (this will let the account access data but not change it):

Service accounts – IAM & Admin

console.cloud.google.com/iam-admin/serviceaccounts

Google Cloud Platform My First Project

IAM & Admin Service accounts + CREATE SERVICE ACCOUNT

Service accounts for project "My First Project"

A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. [Learn more about service accounts.](#)

Organization policies can be used to secure service accounts and block risky service account features, such as automatic IAM Grants, key creation/upload, or the creation of service accounts entirely. [Learn more about service account organization policies.](#)

Filter Enter property name or value

Email	Status	Name ↑	Description	Ke	Actions
No rows to display					

Labels Tags Settings Privacy & Security Manage resources

Service Accounts

Labels Tags Settings Privacy & Security Manage resources

Service Accounts

Create service account – IAM

console.cloud.google.com/iam-admin/serviceaccounts

Google Cloud Platform My First Project

IAM & Admin Create service account

1 Service account details

Service account name

Display name for this service account

Service acco...  X C

Service account description

Describe what this service account will do

CREATE

2 Grant this service account access to project (optional)

3 Grant users access to this service account (optional)

The screenshot shows the 'Create service account' page in the Google Cloud Platform. On the left, a sidebar lists various IAM-related options: IAM, Identity & Organization, Policy Troubleshooter, Policy Analyzer, Organization Policies, Service Accounts (which is selected and highlighted in blue), Labels, Tags, Settings, Privacy & Security, and Identity-Aware Proxy. The main content area is titled 'Create service account' and contains three numbered steps. Step 1, 'Service account details', is completed with a checkmark. Step 2, 'Grant this service account access to project (optional)', shows a 'Role' dropdown set to 'Viewer' with a note 'Read access to all resources.' and a 'Condition' link. A 'CONTINUE' button is below this. Step 3, 'Grant users access to this service account (optional)', is currently empty. At the bottom are 'DONE' and 'CANCEL' buttons.

*push\_pin*

#### Note

The button **CREATE SERVICE ACCOUNT** is gray, you don't have the correct permissions. Ask the admin of your Google Cloud project for help.

After clicking **DONE**, you should be back on the service accounts overview. First, note down the email address of the account you just created (**important for next step!**). Then, create a JSON key file for the new account and download it:

Service accounts – IAM & Admin

console.cloud.google.com/iam-admin/serviceaccounts

Google Cloud Platform My First Project

IAM & Admin Service accounts + CREATE SERVICE ACCOUNT DELETE

IAM Identity & Organization Policy Troubleshooter Policy Analyzer Organization Policies Service Accounts Labels Tags Settings Privacy & Security Manage resources

Service accounts for project "My First Projec"

A service account represents a Google Cloud service identity, such as VMs, App Engine apps, or systems running outside Google. [Learn more](#)

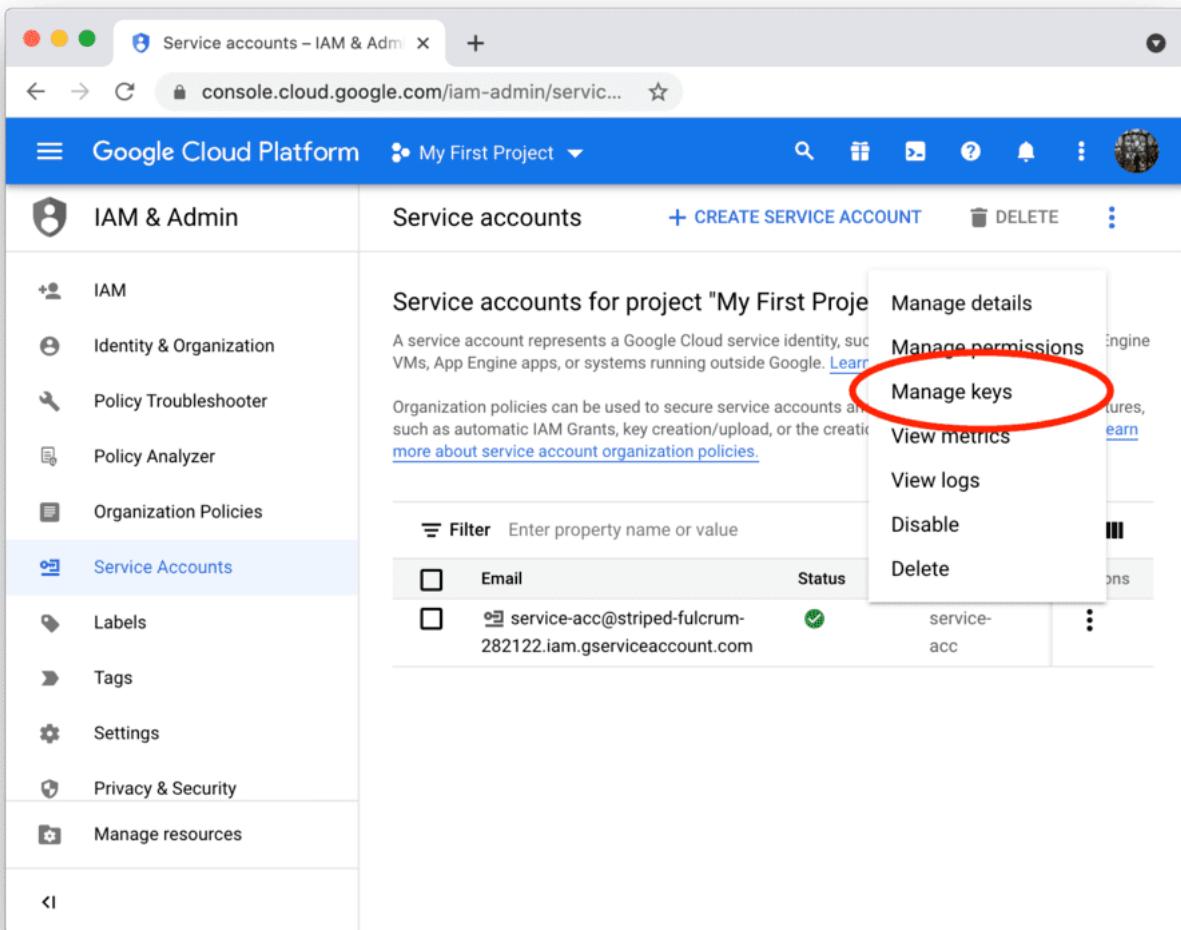
Organization policies can be used to secure service accounts and keys, such as automatic IAM Grants, key creation/upload, or the creation of new service accounts. [Learn more about service account organization policies.](#)

Filter Enter property name or value

Email	Status
service-acc@striped-fulcrum-282122.iam.gserviceaccount.com	Active

Manage details Manage permissions Manage keys View metrics View logs Disable Delete

service-acc



service-acc – IAM & Admin

console.cloud.google.com/iam-admin/serviceaccounts

Google Cloud Platform My First Project

IAM & Admin service-acc DETAILS PERMISSIONS KEYS METRICS LOGS

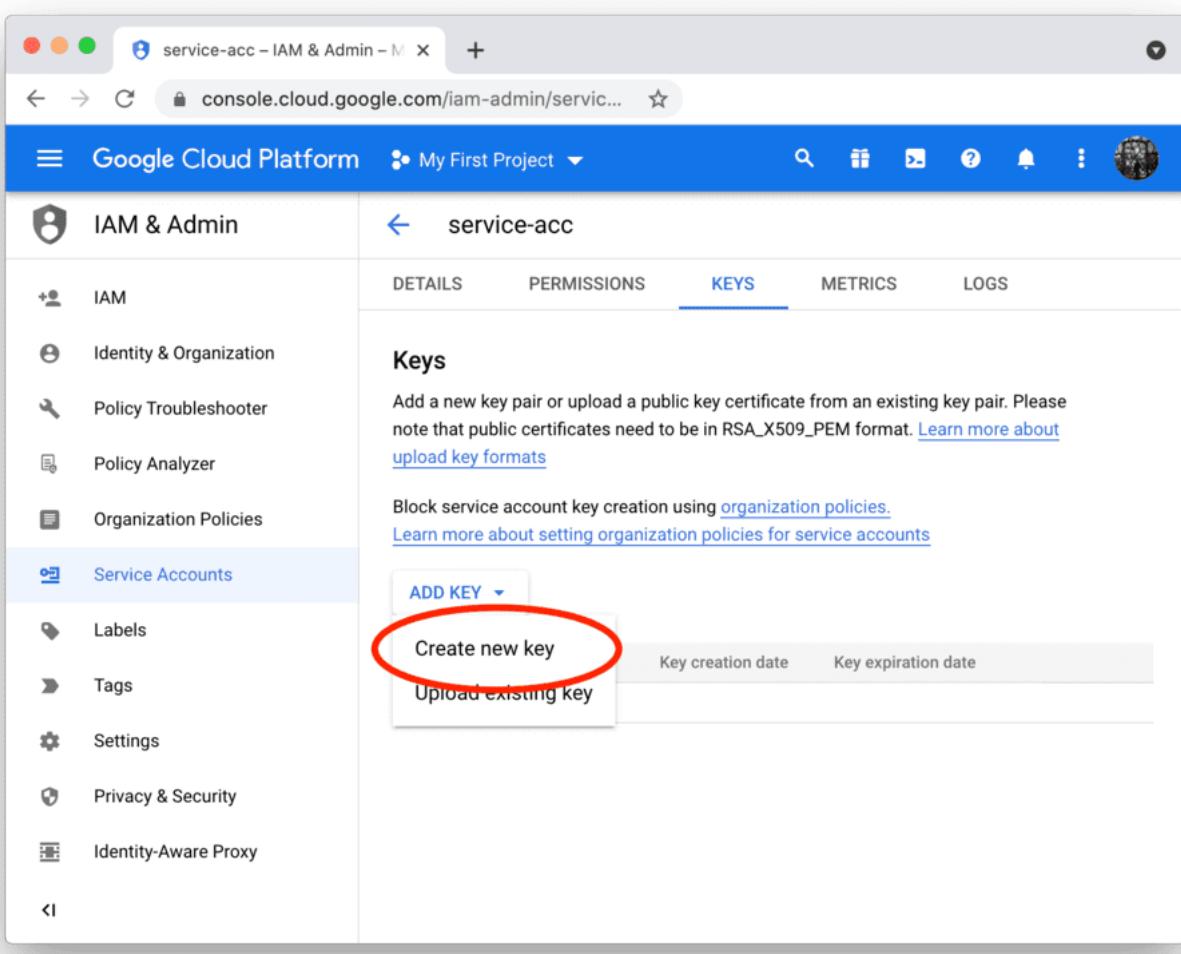
IAM Identity & Organization Policy Troubleshooter Policy Analyzer Organization Policies Service Accounts Labels Tags Settings Privacy & Security Identity-Aware Proxy

Keys

Add a new key pair or upload a public key certificate from an existing key pair. Please note that public certificates need to be in RSA\_X509\_PEM format. [Learn more about upload key formats](#)

Block service account key creation using [organization policies](#). [Learn more about setting organization policies for service accounts](#)

ADD KEY Create new key Upload existing key Key creation date Key expiration date



The screenshot shows the Google Cloud Platform interface for managing service accounts. On the left, a sidebar lists various options like IAM, Identity & Organization, Policy Troubleshooter, and Service Accounts. The 'Service Accounts' option is selected. In the main area, a modal dialog is open titled 'Create private key for "service-acc"'. The dialog contains instructions about downloading a private key file and its security. It offers two key types: 'JSON' (selected) and 'P12'. Below the key type selection, there are 'Recommended' and 'For backward compatibility with code using the P12 format' options. At the bottom right of the dialog are 'CANCEL' and 'CREATE' buttons.

## Share the Google Sheet with the service account

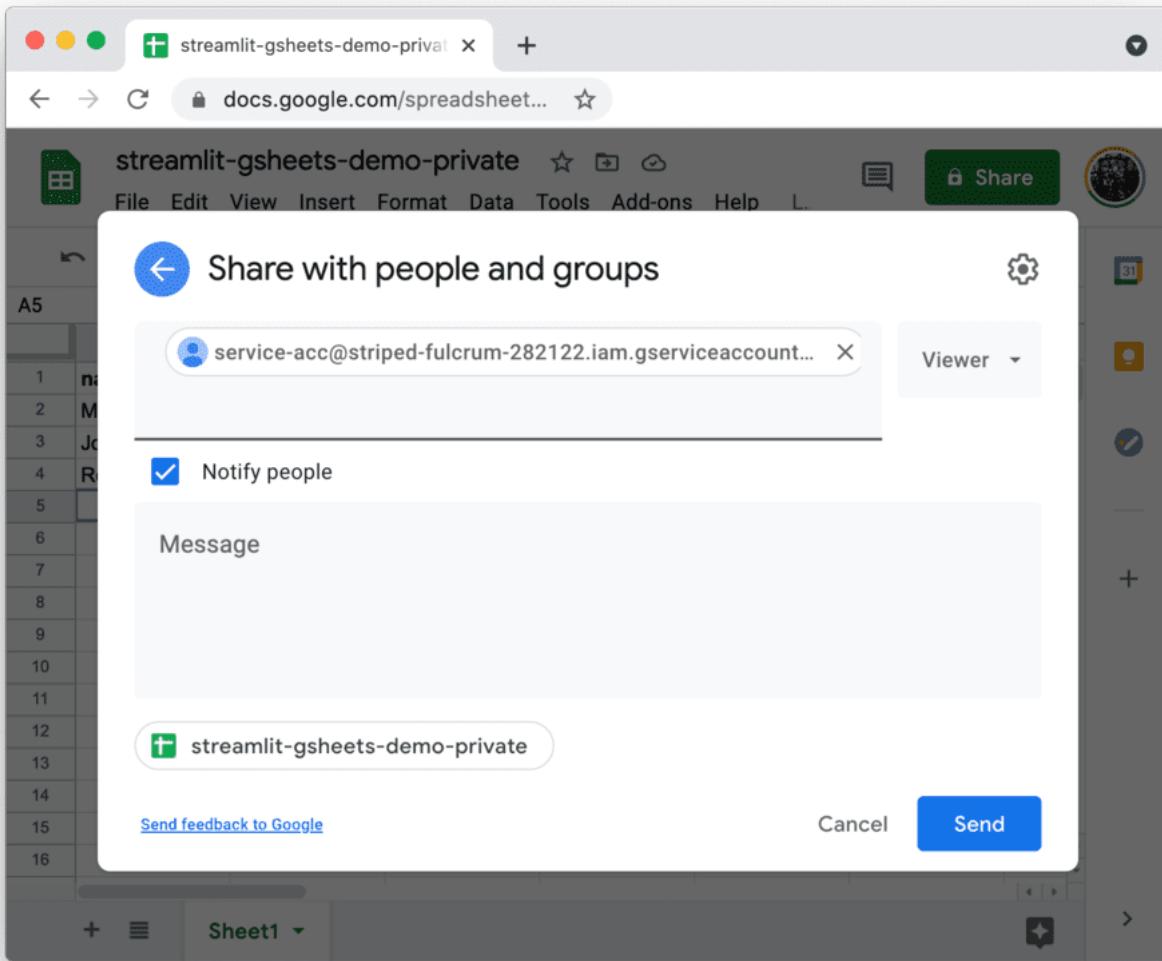
8

By default, the service account you just created cannot access your Google Sheet. To give it access, click on the **Share** button in the Google Sheet, add the email of the service account (noted down in step 2), and choose the correct permission (if you just want to read the data, **Viewer** is enough):

A screenshot of a Google Sheets document titled "streamlit-gsheets-demo-private". The document contains a single sheet named "Sheet1" with the following data:

	A	B	C	D	E	F
1	<b>name</b>	<b>pet</b>				
2	Mary	dog				
3	John	cat				
4	Robert	bird				
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						

The "Share" button in the top right corner is highlighted with a red circle.



## Add the key file to your local app secrets

8

Your local Streamlit app will read secrets from a file `.streamlit/secrets.toml` in your app's root directory. Create this file if it doesn't exist yet and add the URL of your Google Sheet plus the content of the key file you downloaded to it as shown below:

```
# .streamlit/secrets.toml

private_gsheets_url = "https://docs.google.com/spreadsheets/d/12345/edit?usp=sharing"

[gcp_service_account]
type = "service_account"
project_id = "xxx"
private_key_id = "xxx"
private_key = "xxx"
client_email = "xxx"
client_id = "xxx"
auth_uri = "https://accounts.google.com/o/oauth2/auth"
token_uri = "https://oauth2.googleapis.com/token"
auth_provider_x509_cert_url = "https://www.googleapis.com/oauth2/v1/certs"
client_x509_cert_url = "xxx"

priority_high
```

### Important

Add this file to `.gitignore` and don't commit it to your Github repo!

## Copy your app secrets to the cloud

8

As the `secrets.toml` file above is not committed to Github, you need to pass its content to your deployed app (on Streamlit Cloud) separately. Go to the [app dashboard](#) and in the app's dropdown menu, click on **Edit Secrets**. Copy the content of `secrets.toml` into the text area. More information is available at [Secrets Management](#).

The screenshot shows a Streamlit app settings page. On the left, there's a sidebar with 'App settings', 'Sharing', and 'Secrets' (which is currently selected). The main area is titled 'Secrets' and contains instructions about providing environment variables and secrets using TOML format. It shows a code editor with the following TOML configuration:

```
DB_USERNAME = "myuser"  
DB_TOKEN = "abcdef"  
  
[some_section]  
some_key = 1234
```

A blue 'Save' button is located at the bottom right of the editor.

## Add gsheetsdb to your requirements file

8

Add the [gsheetsdb](#) package to your `requirements.txt` file, preferably pinning its version (replace `x.x.x` with the version you want installed):

```
# requirements.txt  
gsheetsdb==x.x.x
```

## Write your Streamlit app

8

Copy the code below to your Streamlit app and run it.

```
# streamlit_app.py  
  
import streamlit as st  
from google.oauth2 import service_account  
from gsheetsdb import connect  
  
# Create a connection object.  
credentials = service_account.Credentials.from_service_account_info(  
    st.secrets["gcp_service_account"],  
    scopes=[  
        "https://www.googleapis.com/auth/spreadsheets",  
    ],  
)  
conn = connect(credentials=credentials)  
  
# Perform SQL query on the Google Sheet.  
# Uses st.cache to only rerun when the query changes or after 10 min.  
@st.cache(ttl=600)  
def run_query(query):  
    rows = conn.execute(query, headers=1)  
    rows = rows.fetchall()  
    return rows  
  
sheet_url = st.secrets["private_gsheets_url"]  
rows = run_query(f'SELECT * FROM "{sheet_url}"')
```

```
# Print results.  
for row in rows:  
    st.write(f"{{row.name}} has a :{{row.pet}}:")
```

See `st.cache` above? Without it, Streamlit would run the query every time the app reruns (e.g. on a widget interaction). With `st.cache`, it only runs when the query changes or after 10 minutes (that's what `ttl` is for). Watch out: If your database updates more frequently, you should adapt `ttl` or remove caching so viewers always see the latest data. Read more about caching [here](#).

If everything worked out (and you used the example table we created above), your app should look like this:

Mary has a 🐶  
John has a 🐱  
Robert has a 🐈

Made with Streamlit

Was this page helpful?

Yes  No

[edit](#)[Suggest edits](#)

[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: Tableau](#) [Next: Public Google Sheet](#) →

[Home](#) [Contact Us](#) [Community](#)





## Documentation



Search

- 

Streamlit library

- Get started



- API reference



- Advanced features



- Components



- Changelog

- Cheat sheet

- 

Streamlit Cloud

- Get started



- Trust and Security

- Release notes

- Troubleshooting

- 

Knowledge base

- Tutorials



- Connect to data sources



- AWS S3
  - BigQuery
  - Snowflake
  - Microsoft SQL Server
  - Firestore 
  - MongoDB
  - MySQL
  - PostgreSQL
  - Tableau
  - Private Google Sheet
  - Public Google Sheet
  - TigerGraph
  - Deta Base
  - Supabase
  - Google Cloud Storage
- Session State basics
- Using Streamlit
  - Streamlit Components
  - Installing dependencies
  - Deployment issues

[Home](#) / [Knowledge base](#) / [Tutorials](#) / [Connect to data sources](#) / [\*\*Public Google Sheet\*\*](#)

# Connect Streamlit to a public Google Sheet

[\*\*Introduction\*\*](#)

This guide explains how to securely access a public Google Sheet from Streamlit Cloud. It uses the gsheetsdbs library and Streamlit's secrets management.

This method requires you to enable link sharing for your Google Sheet. While the sharing link will not appear in your code (and actually acts as sort of a password!), someone with the link can get all the data in the Sheet. If you don't want this, follow the (more complicated) guide Connect Streamlit to a private Google Sheet.

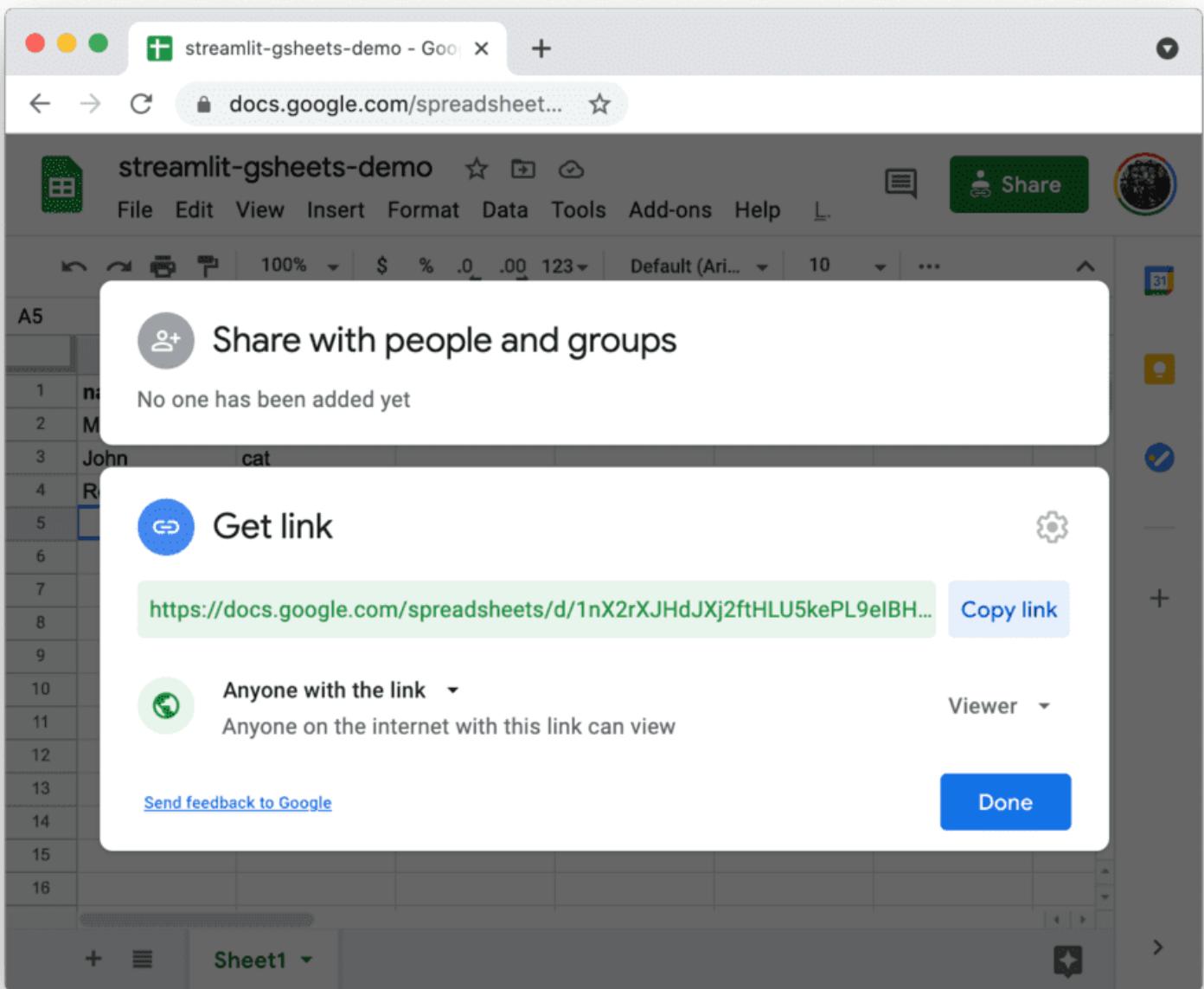
## Create a Google Sheet and turn on link sharing

### Note

If you already have a Sheet that you want to access, feel free to skip to the next step.

The screenshot shows a Google Sheets interface with the following data:

	A	B
1	name	pet
2	Mary	dog
3	John	cat
4	Robert	bird
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		



## Add the Sheets URL to your local app secrets

Your local Streamlit app will read secrets from a file `.streamlit/secrets.toml` in your app's root directory. Create this file if it doesn't exist yet and add the share link of your Google Sheet to it as shown below:

```
# .streamlit/secrets.toml
```

```
public_gsheets_url = "https://docs.google.com/spreadsheets/d/xxxxxxxx/edit#gid=0"
```

! **Important**

Add this file to [.gitignore](#) and don't commit it to your Github repo!

## Copy your app secrets to the cloud

As the `secrets.toml` file above is not committed to Github, you need to pass its content to your deployed app (on Streamlit Cloud) separately. Go to the app dashboard and in the app's dropdown menu, click on **Edit Secrets**. Copy the content of `secrets.toml` into the text area. More information is available at Secrets Management.

The screenshot shows a web browser window for 'awesomestreamlit's apps' on 'share.streamlit.io'. The left sidebar has 'App settings', 'Sharing', and 'Secrets' (which is selected). The main content area is titled 'Secrets' and contains instructions about TOML format and security. A code editor shows the following TOML content:

```
DB_USERNAME = "myuser"  
DB_TOKEN = "abcdef"  
  
[some_section]  
some_key = 1234
```

A blue 'Save' button is at the bottom right of the editor.

## Add gsheetsdb to your requirements file

Add the `gsheetsdb` package to your `requirements.txt` file, preferably pinning its version (replace `x.x.x` with the version you want installed):

```
# requirements.txt
gsheetsdb==x.x.x
```

## Write your Streamlit app

Copy the code below to your Streamlit app and run it.

```
# streamlit_app.py

import streamlit as st
from gsheetsdb import connect

# Create a connection object.
conn = connect()

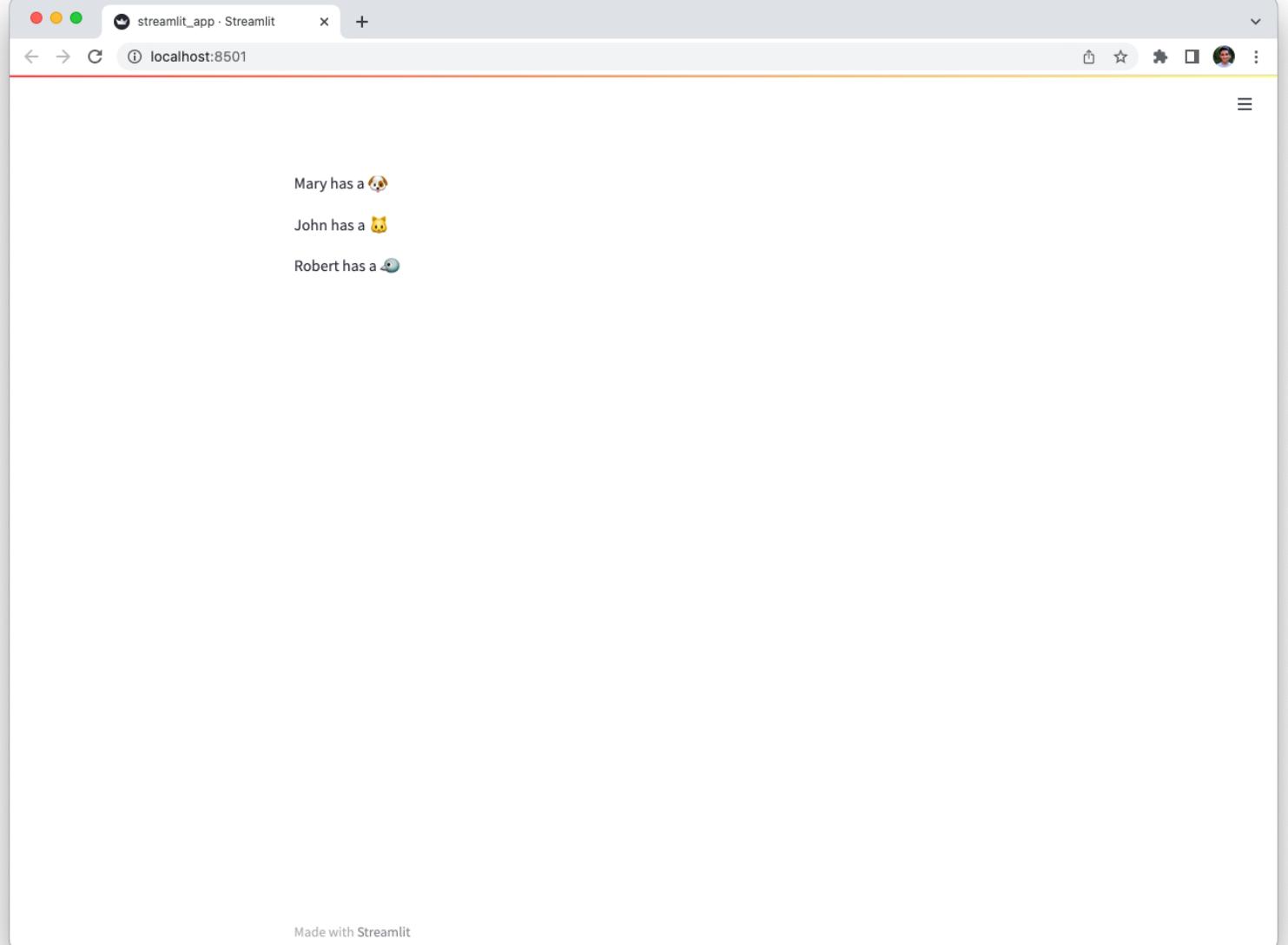
# Perform SQL query on the Google Sheet.
# Uses st.cache to only rerun when the query changes or after 10 min.
@st.cache(ttl=600)
def run_query(query):
    rows = conn.execute(query, headers=1)
    rows = rows.fetchall()
    return rows

sheet_url = st.secrets["public_gsheets_url"]
rows = run_query(f'SELECT * FROM "{sheet_url}"')

# Print results.
for row in rows:
    st.write(f'{row.name} has a :{row.pet}:')
```

See `st.cache` above? Without it, Streamlit would run the query every time the app reruns (e.g. on a widget interaction). With `st.cache`, it only runs when the query changes or after 10 minutes (that's what `ttl` is for). Watch out: If your database updates more frequently, you should adapt `ttl` or remove caching so viewers always see the latest data. Read more about caching [here](#).

If everything worked out (and you used the example table we created above), your app should look like this:



## Was this page helpful?

 Yes

 No

 [Suggest edits](#)



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.





# Publish a Component

## Publish to PyPI

Publishing your Streamlit Component to [PyPI](#) makes it easily accessible to Python users around the world. This step is completely optional, so if you won't be releasing your component publicly, you can skip this section!

### ★ Note

For [static Streamlit Components](#), publishing a Python package to PyPI follows the same steps as the [core PyPI packaging instructions](#). A static Component likely contains only Python code, so once you have your `setup.py` file correct and [generate your distribution files](#), you're ready to [upload to PyPI](#).

[Bi-directional Streamlit Components](#) at minimum include both Python and JavaScript code, and as such, need a bit more preparation before they can be published on PyPI. The remainder of this page focuses on the bi-directional Component preparation process.

## Prepare your Component

A bi-directional Streamlit Component varies slightly from a pure Python library in that it must contain pre-compiled frontend code. This is how base Streamlit works as well; when you `pip install streamlit`, you are getting a Python library where the HTML and frontend code contained within it have been compiled into static assets.

The [component-template](#) GitHub repo provides the folder structure necessary for PyPI publishing. But before you can publish, you'll need to do a bit of housekeeping:

1. Give your Component a name, if you haven't already

- Rename the `template/my_component/` folder to `template/<component_name>/`
- Pass your component's name as the the first argument to `declare_component()`

2. Edit `MANIFEST.in`, change the path for recursive-include from `package/frontend/build *` to `<component name>/frontend/build *`

3. Edit `setup.py`, adding your component's name and other relevant info

4. Create a release build of your frontend code. This will add a new directory, `frontend/build/`, with your compiled frontend in it:

```
$ cd frontend  
$ npm run build
```

5. Pass the build folder's path as the `path` parameter to `declare_component`. (If you're using the template Python file, you can set `_RELEASE = True` at the top of the file):

```
import streamlit.components.v1 as components  
  
# Change this:  
# component = components.declare_component("my_component", url="http://lo  
  
# To this:  
parent_dir = os.path.dirname(os.path.abspath(__file__))  
build_dir = os.path.join(parent_dir, "frontend/build")  
component = components.declare_component("new_component_name", path=buil
```

## Build a Python wheel

Once you've changed the default `my_component` references, compiled the HTML and JavaScript code and set your new component name in `components.declare_component()`, you're ready to build a Python wheel:

1. Make sure you have the latest versions of setuptools, wheel, and twine

2. Create a wheel from the source code:

```
# Run this from your component's top-level directory; that is,  
# the directory that contains `setup.py`  
$ python setup.py sdist bdist_wheel
```

# Upload your wheel to PyPI

With your wheel created, the final step is to upload to PyPI. The instructions here highlight how to upload to [Test PyPI](#), so that you can learn the mechanics of the process without worrying about messing anything up. Uploading to PyPI follows the same basic procedure.

## 1. Create an account on [Test PyPI](#) if you don't already have one

- Visit <https://test.pypi.org/account/register/> and complete the steps
- Visit <https://test.pypi.org/manage/account/#api-tokens> and create a new API token. Don't limit the token scope to a particular project, since you are creating a new project. Copy your token before closing the page, as you won't be able to retrieve it again.

## 2. Upload your wheel to Test PyPI. [twine](#) will prompt you for a username and password. For the username, use **token**. For the password, use your token value from the previous step, including the [pypi-](#) prefix:

```
python3 -m twine upload --repository testpypi dist/*
```

## 3. Install your newly-uploaded package in a new Python project to make sure it works:

```
python -m pip install --index-url https://test.pypi.org/simple/ --no-deps <your-component-name>
```

If all goes well, you're ready to upload your library to PyPI by following the instructions at <https://packaging.python.org/tutorials/packaging-projects/#next-steps>.

Congratulations, you've created a publicly-available Streamlit Component!

## Promote your Component!

We'd love to help you share your Component with the Streamlit Community! To share it, please post on the [Streamlit 'Show the Community!' Forum category](#) with the title similar to "New Component: <[your component name](#)>, a new way to do X".

You can also Tweet at us [@streamlit](#) so that we can retweet your announcement for you.

If you host your code on GitHub, add the tag [streamlit-component](#), so that it's listed in the [GitHub streamlit-component topic](#):



Search or jump to...

Pull requests Issues Marketplace Explore

+

snehanekre / your-component

Private

Unwatch

1

▼

Fork

0

Star

0

▼

Code

Issues

Pull requests

Actions

Projects

Security

Insights

Settings

main

1 branch

0 tags

Go to file

Add file

Code

snehanekre	Create setup.py	0f3e616 10 minutes ago	3 commits
your_component	Create __init__.py	10 minutes ago	
README.md	Initial commit	11 minutes ago	
setup.py	Create setup.py	10 minutes ago	

README.md

## your-component

This is an example Streamlit component.

### About

This is an example Streamlit component.

Readme

0 stars

1 watching

0 forks

### Releases

No releases published

[Create a new release](#)

### Packages

No packages published

[Publish your first package](#)

### Languages

Python 100.0%

© 2022 GitHub, Inc.

[Terms](#)

[Privacy](#)

[Security](#)

[Status](#)

[Docs](#)

[Contact GitHub](#)

[Pricing](#)

[API](#)

[Training](#)

[Blog](#)

[About](#)

Add the `streamlit-component` tag to your GitHub repo

## Was this page helpful?

Yes

No

[Suggest edits](#)



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.

# Documentation



Search



Streamlit library

- Get started



- API reference



- Advanced features



- Components



- Changelog

- Cheat sheet



Streamlit Cloud

- Get started



- Trust and Security

- Release notes

- Troubleshooting



Knowledge base

- Tutorials



- Using Streamlit

- Streamlit Components

- Installing dependencies
- Deployment issues

[Home](#) / [Knowledge base](#) / [Using Streamlit](#) / [How can I make st.pydeck\\_chart use custom Mapbox styles?](#)

# How can I make st.pydeck\_chart use custom Mapbox styles?

If you are supplying a Mapbox token, but the resulting `pydeck_chart` doesn't show your custom Mapbox styles, please check that you are adding the Mapbox token to the Streamlit `config.toml` configuration file. Streamlit DOES NOT read Mapbox tokens from inside of a PyDeck specification (i.e. from inside of the Streamlit app). Please see this [forum thread](#) for more information.

Was this page helpful?

 Yes     No

 [Suggest edits](#)



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← **Previous:** What is the path of Streamlit's config.toml file?

**Next:** How to record a screencast? →

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.

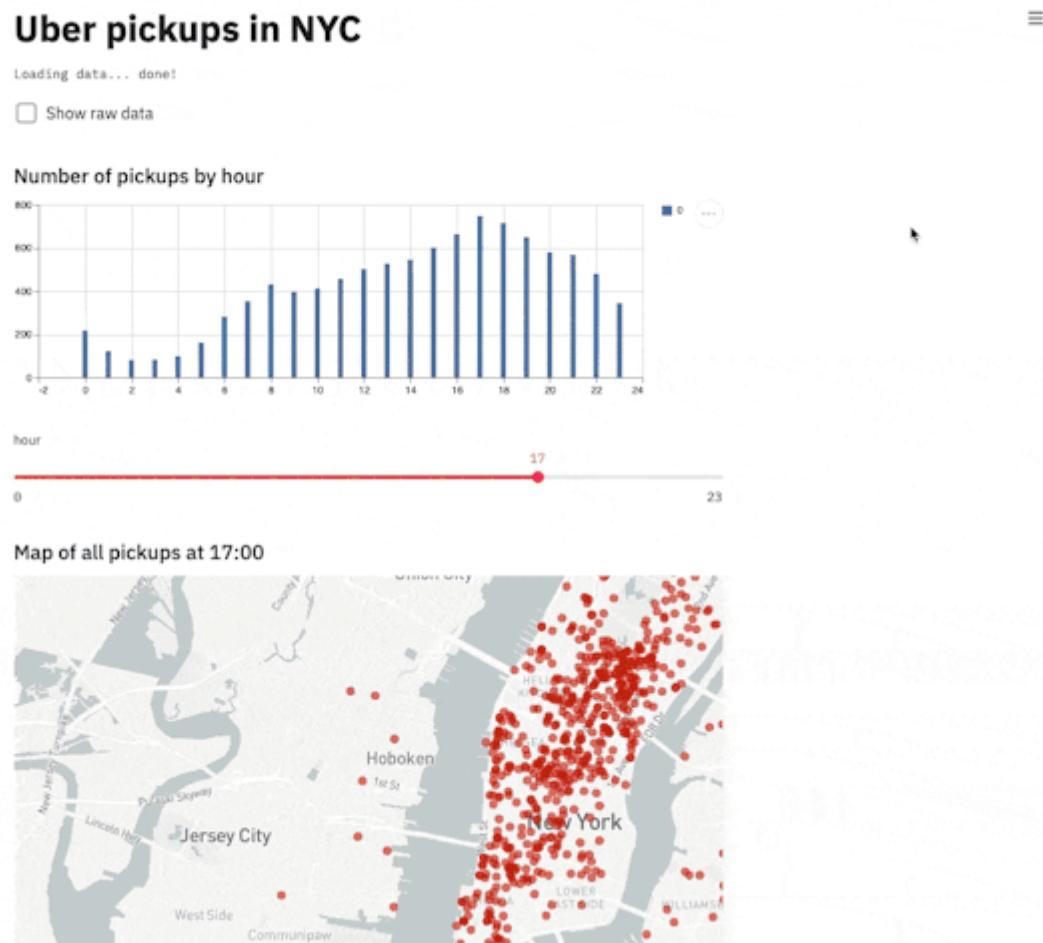


# How to record a screencast?

After you've built a Streamlit app, you may want to discuss some of it with co-workers over email or Slack, or share it with the world on Twitter. A great way to do that is with Streamlit's built-in screencast recorder. With it, you can record, narrate, stop, save, and share with a few clicks.

To start a screencast, locate the menu in the upper right corner of your app ( $\equiv$ ), select **Record a screencast**, and follow the prompts. Before the recording starts, you'll see a countdown — this means it's showtime.

To stop your screencast, go back to the menu ( $\equiv$ ) and select **Stop recording** (or hit the **ESC** key). Follow the prompts to preview your recording and save it to disk. That's it, you're ready to share your Streamlit app.



Was this page helpful?

Yes

No

---

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

← **Previous:** How can I make st.pydeck\_chart use custom Mapbox styles?

**Next:** How to remove ". Streamlit" from the app title? →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# App is not loading when running remotely

Below are a few common errors that occur when users spin up their own solution to host a Streamlit app remotely.

To learn about a deceptively simple way to host Streamlit apps that avoids all the issues below, check out [Streamlit Cloud](#).

## Symptom #1: The app never loads

When you enter the app's URL in a browser and all you see is a **blank page**, a "**Page not found**" error, a "**Connection refused**" error, or anything like that, first check that Streamlit is actually running on the remote server. On a Linux server you can SSH into it and then run:

```
ps -Al | grep streamlit
```

If you see Streamlit running, the most likely culprit is the Streamlit port not being exposed. The fix depends on your exact setup. Below are three example fixes:

- **Try port 80:** Some hosts expose port 80 by default. To set Streamlit to use that port, start Streamlit with the `--server.port` option:

```
streamlit run my_app.py --server.port=80
```

- **AWS EC2 server:** First, click on your instance in the [AWS Console](#). Then scroll down and click on *Security Groups* → *Inbound* → *Edit*. Next, add a *Custom TCP rule* that allows the *Port Range* `8501` with Source `0.0.0.0/0`.
- **Other types of server:** Check the firewall settings.

If that still doesn't solve the problem, try running a simple HTTP server instead of Streamlit, and seeing if *that* works correctly. If it does, then you know the problem lies somewhere in your Streamlit app or configuration (in which case you should ask for help in our [forums!](#)) If not, then it's definitely unrelated to Streamlit.

How to start a simple HTTP server:

```
python -m http.server [port]
```

## Symptom #2: The app says "Please wait..." forever

If when you try to load your app in a browser you see a blue box in the center of the page with the text "Please wait...", the underlying cause is likely one of the following:

- Misconfigured [CORS](#) protection.
- Server is stripping headers from the Websocket connection, thereby breaking compression.

To diagnose the issue, try temporarily disabling CORS protection by running Streamlit with the [--server.enableCORS](#) flag set to [false](#):

```
streamlit run my_app.py --server.enableCORS=false
```

If this fixes your issue, **you should re-enable CORS protection** and then set [browser.serverPort](#) and [browser.serverAddress](#) to the URL and port of your Streamlit app.

If the issue persists, try disabling websocket compression by running Streamlit with the [--server.enableWebSocketCompression](#) flag set to [false](#)

```
streamlit run my_app.py --server.enableWebSocketCompression=false
```

If this fixes your issue, your server setup is likely stripping the [Sec-WebSocket-Extensions](#) HTTP header that is used to negotiate Websocket compression.

Compression is not required for Streamlit to work, but it's strongly recommended as it improves performance. If you'd like to turn it back on, you'll need to find which part of your infrastructure is stripping the [Sec-WebSocket-Extensions](#) HTTP header and change that behavior.

# Symptom #3: Unable to upload files when running in multiple replicas

If the file uploader widget returns an error with status code 403, this is probably due to a misconfiguration in your app's XSRF protection logic.

To diagnose the issue, try temporarily disabling XSRF protection by running Streamlit with the `--server.enableXsrfProtection` flag set to `false`:

```
streamlit run my_app.py --server.enableXsrfProtection=false
```

If this fixes your issue, **you should re-enable XSRF protection** and then configure your app to use the same secret across every replica by setting the `server.cookieSecret` config option to the same hard-to-guess string everywhere.

## Was this page helpful?

 Yes     No

 [Suggest edits](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← **Previous:** Organizing your apps with workspaces on Streamlit Cloud

**Next:** Argh. This app has gone over its resource limits →





# How to remove ". Streamlit" from the app title?

Using `st.set_page_config` to assign the page title will not append ". Streamlit" to that title. E.g.:

```
import streamlit as st

st.set_page_config(
    page_title="Ex-stream-ly Cool App",
    page_icon="💻",
    layout="wide",
    initial_sidebar_state="expanded",
)
```



**Was this page helpful?**

Yes

No

[Suggest edits](#)



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

**Next:** How do you retrieve the filename of a file uploaded with st.file\_uploader? →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# Argh. This app has gone over its resource limits

Sorry! It means you've hit the [resource limits](#) of your [Streamlit Cloud](#) account. One way to avoid this is to [upgrade your plan](#) to one with higher resource limits. But there are also a few things you can change in your app to make it less resource-hungry:

- Reboot your app (temporary fix)
- Use `st.experimental_memo` or `st.experimental_singleton` to load models or data only once
- Restrict the cache size with `ttl` or `max_entries`
- Move big datasets to a database
- Profile your app's memory usage

Check out our [blog post](#) on “[Common app problems: Resource limits](#)” for more in-depth tips prevent your app from hitting the [resource limits](#) of the Streamlit Cloud.

Related forum posts:

- <https://discuss.streamlit.io/t/common-app-problems-resource-limits/16969>
- <https://blog.streamlit.io/common-app-problems-resource-limits/>

Was this page helpful?

Yes    No

[Suggest edits](#)



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[← Previous:](#) App is not loading when running remotely

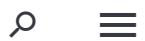
**Next:** How do I share apps with viewers outside my organization? [→](#)

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



How do you retrieve the filename of a file uploaded with st.file\_uploader?

# How do you retrieve the filename of a file uploaded with st.file\_uploader?

If you upload a single file (i.e. `accept_multiple_files=False`), the filename can be retrieved by using the `.name` attribute on the returned UploadedFile object:

```
import streamlit as st

uploaded_file = st.file_uploader("Upload a file")

if uploaded_file:
    st.write("Filename: ", uploaded_file.name)
```

If you upload multiple files (i.e. `accept_multiple_files=True`), the individual filenames can be retrieved by using the `.name` attribute on each UploadedFile object in the returned list:

```
import streamlit as st

uploaded_files = st.file_uploader("Upload multiple files", accept_multiple_files=True)

if uploaded_files:
    for uploaded_file in uploaded_files:
        st.write("Filename: ", uploaded_file.name)
```

Related forum posts:

- <https://discuss.streamlit.io/t/is-it-possible-to-get-uploaded-file-file-name/7586>

Was this page helpful?

Yes

No

 Suggest edits

---

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

← **Previous:** How to remove ". Streamlit" from the app title?

**Next:** Sanity checks →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# Sanity checks

If you're having problems running your Streamlit app, here are a few things to try out.

## Check #0: Are you using a Streamlit-supported version of Python?

Streamlit will maintain backwards-compatibility with earlier Python versions as practical, guaranteeing compatibility with *at least* the last three minor versions of Python 3.

As new versions of Python are released, we will try to be compatible with the new version as soon as possible, though frequently we are at the mercy of other Python packages to support these new versions as well.

Streamlit currently supports versions 3.7, 3.8, 3.9, and 3.10 of Python.

## Check #1: Is Streamlit running?

On a Mac or Linux machine, type this on the terminal:

```
ps -Al | grep streamlit
```



If you don't see `streamlit run` in the output (or `streamlit hello`, if that's the command you ran) then the Streamlit server is not running. So re-run your command and see if the bug goes away.

## Check #2: Is this an already-fixed Streamlit bug?

We try to fix bugs quickly, so many times a problem will go away when you upgrade Streamlit. So the first thing to try when having an issue is upgrading to the latest version of Streamlit:

```
pip install --upgrade streamlit  
streamlit version
```



...and then verify that the version number printed corresponds to the version number displayed on [PyPI](#).

Try reproducing the issue now. If not fixed, keep reading on.

## Check #3: Are you running the correct Streamlit binary?

Let's check whether your Python environment is set up correctly. Edit the Streamlit script where you're experiencing your issue, **comment everything out, and add these lines instead:**

```
import streamlit as st  
st.write(st.__version__)
```



...then call `streamlit run` on your script and make sure it says the same version as above. If not the same version, check out [these instructions](#) for some sure-fire ways to set up your environment.

## Check #4: Is your browser caching your app too aggressively?

There are two easy ways to check this:

1. Load your app in a browser then press `Ctrl-Shift-R` or `⌘-Shift-R` to do a hard refresh (Chrome/Firefox).
2. As a test, run Streamlit on another port. This way the browser starts the page with a brand new cache. For that, pass the `--server.port` argument to Streamlit on the command line:

```
streamlit run my_app.py --server.port=9876
```



## Check #5: Is this a Streamlit regression?

If you've upgraded to the latest version of Streamlit and things aren't working, you can downgrade at any time using this command:

```
pip install --upgrade streamlit==1.0.0
```



...where **1.0.0** is the version you'd like to downgrade to. See [Changelog](#) for a complete list of Streamlit versions.

## Check #6 [Windows]: Is Python added to your PATH?

When installed by downloading from [python.org](#), Python is not automatically added to the [Windows system PATH](#). Because of this, you may get error messages like the following:

Command Prompt:

```
C:\Users\streamlit> streamlit hello  
'streamlit' is not recognized as an internal or external command,  
operable program or batch file.
```



PowerShell:

```
PS C:\Users\streamlit> streamlit hello  
streamlit : The term 'streamlit' is not recognized as the name of a cmdlet, fu  
the path is correct and try again.  
At line:1 char:1  
+ streamlit hello  
+ ~~~~~~  
+ CategoryInfo          : ObjectNotFound: (streamlit:String) [], CommandNo  
+ FullyQualifiedErrorId : CommandNotFoundException
```



To resolve this issue, add [Python to the Windows system PATH](#).

After adding Python to your Windows PATH, you should then be able to follow the instructions in our [Get Started](#) section.

## Check #7 [Windows]: Do you need Build Tools for Visual Studio installed?

Streamlit includes [pyarrow](#) as an install dependency. Occasionally, when trying to install Streamlit from PyPI, you may see errors such as the following:

```
Using cached pyarrow-1.0.1.tar.gz (1.3 MB)
Installing build dependencies ... error
ERROR: Command errored out with exit status 1:
command: 'c:\users\streamlit\appdata\local\programs\python\python38-32\pyt
  cwd: None
```

Complete output (319 lines):

```
Running setup.py install for numpy: finished with status 'error'
ERROR: Command errored out with exit status 1:
command: 'c:\users\streamlit\appdata\local\programs\python\python38-32\pyt
  cwd: C:\Users\streamlit\AppData\Local\Temp\pip-install-0jwfwx_u\nur
Complete output (298 lines):
```

```
blas_opt_info:
blas_mkl_info:
No module named 'numpy.distutils._msvccompiler' in numpy.distutils; try:
customize MSVCCCompiler
libraries mkl_rt not found in ['c:\\users\\streamlit\\appdata\\local\\'
NOT AVAILABLE
```

```
blas_info:
No module named 'numpy.distutils._msvccompiler' in numpy.distutils; try:
customize MSVCCCompiler
libraries blas not found in ['c:\\users\\streamlit\\appdata\\local\\'
NOT AVAILABLE
```

# <truncated for brevity> #

```
c:\users\streamlit\appdata\local\programs\python\python38-32\lib\distut:
  warnings.warn(msg)
running install
running build
running config_cc
unifing config_cc, config, build_clib, build_ext, build commands --comp:
running config_fc
unifing config_fc, config, build_clib, build_ext, build commands --fcomp:
running build_src
build_src
```

```
building py_modules sources
creating build
creating build\src.win32-3.8
creating build\src.win32-3.8\numpy
creating build\src.win32-3.8\numpy\distutils
building library "npymath" sources
No module named 'numpy.distutils._msvccompiler' in numpy.distutils; try:
error: Microsoft Visual C++ 14.0 is required. Get it with "Build Tools" .
```

---

ERROR: Command errored out with `exit` status 1: 'c:\users\streamlit\appdata\'

---

This error indicates that Python is trying to compile certain libraries during install, but it cannot find the proper compilers on your system, as reflected by the line **error: Microsoft Visual C++ 14.0 is required. Get it with "Build Tools for Visual Studio" .**

Installing Build Tools for Visual Studio should resolve this issue.

## Was this page helpful?

 Yes     No

 Suggest edits

## Still have questions?

Our forums are full of helpful information and Streamlit experts.

---

 **Previous:** How do you retrieve the filename of a file uploaded with st.file\_uploader?

**Next:** How can I make Streamlit watch for changes in other modules I'm importing in my app? 



Copyright © 2022, Streamlit Inc.



# Documentation



Search

- 

Streamlit library

- Get started



- API reference



- Write and magic



- Text elements



- Data display elements



- Chart elements



- Input widgets



- Media elements



- Layouts and containers



- Status elements



- Control flow



- Utilities



- Mutate charts
  - State management
  - Performance
- +

- Advanced features

+

- Components

+

- Changelog

- Cheat sheet

- 

## Streamlit Cloud

- Get started

+

- Trust and Security

- Release notes 

- Troubleshooting

- 

## Knowledge base

- Tutorials

+

- Using Streamlit

- Streamlit Components

- Installing dependencies

- Deployment issues

Session State is a way to share variables between reruns, for each user session. In addition to the ability to store and persist state, Streamlit also exposes the ability to manipulate state using Callbacks.

Check out this Session State basics tutorial video by Streamlit Developer Advocate Dr. Marisa Smith to get started:

Session State basics



Copy link



## Initialize values in Session State

The Session State API follows a field-based API, which is very similar to Python dictionaries:

```
# Initialization
if 'key' not in st.session_state:
    st.session_state['key'] = 'value'

# Session State also supports attribute based syntax
if 'key' not in st.session_state:
    st.session_state.key = 'value'
```

## Reads and updates

Read the value of an item in Session State and display it by passing to `st.write` :

```
# Read  
st.write(st.session_state.key)
```

```
# Outputs: value
```

Update an item in Session State by assigning it a value:

```
st.session_state.key = 'value2'      # Attribute API  
st.session_state['key'] = 'value2'   # Dictionary like API
```

Curious about what is in Session State? Use `st.write` or magic:

```
st.write(st.session_state)  
  
# With magic:  
st.session_state
```

Streamlit throws a handy exception if an uninitialized variable is accessed:

```
st.write(st.session_state['value'])  
  
# Throws an exception!
```

**KeyError: 'st.session\_state has no key "value". Did you forget to initialize it?'**

Traceback:

```
File "/.local/share/lib/python3.8/site-packages/streamlit/script_runner.py", line 34
  exec(code, module.__dict__)
File "/home/app.py", line 3, in <module>
  st.write(st.session_state['value'])
File "/.local/share/lib/python3.8/site-packages/streamlit/state/session_state.py", l
  return state[key]
File "/.local/share/lib/python3.8/site-packages/streamlit/state/session_state.py", l
  raise KeyError(_missing_key_error_message(key))
```

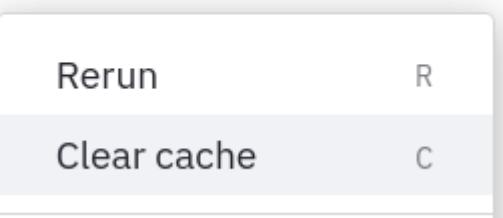
## Delete items

Delete items in Session State using the syntax to delete items in any Python dictionary:

```
# Delete a single key-value pair
del st.session_state[key]

# Delete all the items in Session state
for key in st.session_state.keys():
    del st.session_state[key]
```

Session State can also be cleared by going to Settings → Clear Cache, followed by Rerunning the app.



## Session State and Widget State association

Every widget with a key is automatically added to Session State:

```
st.text_input("Your name", key="name")  
  
# This exists now:  
st.session_state.name
```

## Use Callbacks to update Session State

A callback is a python function which gets called when an input widget changes.

**Order of execution:** When updating Session state in response to **events**, a callback function gets executed first, and then the app is executed from top to bottom.

Callbacks can be used with widgets using the parameters `on_change` (or `on_click`), `args`, and `kwargs`:

### Parameters

- **on\_change** or **on\_click** - The function name to be used as a callback
- **args (tuple)** - List of arguments to be passed to the callback function
- **kwargs (dict)** - Named arguments to be passed to the callback function

Widgets which support the `on_change` event:

- `st.checkbox`
- `st.color_picker`
- `st.date_input`
- `st.multiselect`
- `st.number_input`
- `st.radio`
- `st.select_slider`
- `st.selectbox`
- `st.slider`
- `st.text_area`

- `st.text_input`
- `st.time_input`
- `st.file_uploader`

Widgets which support the `on_click` event:

- `st.button`
- `st.download_button`
- `st.form_submit_button`

To add a callback, define a callback function **above** the widget declaration and pass it to the widget via the `on_change` (or `on_click`) parameter.

## Forms and Callbacks

Widgets inside a form can have their values be accessed and set via the Session State API.

`st.form_submit_button` can have a callback associated with it. The callback gets executed upon clicking on the submit button. For example:

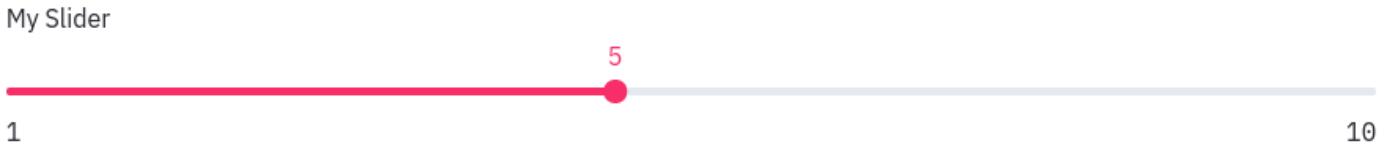
```
def form_callback():
    st.write(st.session_state.my_slider)
    st.write(st.session_state.my_checkbox)

with st.form(key='my_form'):
    slider_input = st.slider('My slider', 0, 10, 5, key='my_slider')
    checkbox_input = st.checkbox('Yes or No', key='my_checkbox')
    submit_button = st.form_submit_button(label='Submit', on_click=form_callback)
```

## Caveats and limitations

- Only the `st.form_submit_button` has a callback in forms. Other widgets inside a form are not allowed to have callbacks.
- `on_change` and `on_click` events are only supported on input type widgets.
- Modifying the value of a widget via the Session state API, after instantiating it, is not allowed and will raise a `StreamlitAPIException`. For example:

```
slider = st.slider(  
    label='My Slider', min_value=1,  
    max_value=10, value=5, key='my_slider')  
  
st.session_state.my_slider = 7  
  
# Throws an exception!
```



**StreamlitAPIException:** `st.session_state.my_slider` cannot be modified after the widget with key `my_slider` is instantiated.

Traceback:

```
File "/home/app.py", line 7, in <module>  
    st.session_state.my_slider = 7
```

- Setting the widget state via the Session State API and using the `value` parameter in the widget declaration is not recommended, and will throw a warning on the first run. For example:

```
st.session_state.my_slider = 7  
  
slider = st.slider(  
    label='Choose a Value', min_value=1,  
    max_value=10, value=5, key='my_slider')
```

The widget with key "my\_slider" was created with a default value but also had its value set via the Session State API.

Choose a Value



- Setting the state of button-like widgets: `st.button`, `st.download_button`, and `st.file_uploader` via the Session State API is not allowed. Such type of widgets are by default `False` and have ephemeral `True` states which are only valid for a single run. For example:

```
if 'my_button' not in st.session_state:  
    st.session_state.my_button = True  
  
st.button('My button', key='my_button')  
  
# Throws an exception!
```

**StreamlitAPIException:** Values for the `st.button` and `st.file_uploader` widgets cannot be set using `st.session_state`.

Traceback:

```
File "/home/app.py", line 7, in <module>  
    st.button('Submit', key='my_button')
```

Was this page helpful?

Yes    No

Suggest edits



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

[← Previous: Mutate charts](#)

[Next: Performance →](#)

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.

## Documentation

*search*

Search

- [description](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [cloud](#)

### [Streamlit Cloud](#)

- [Get started](#)  
*remove*
  - [Deploy an app](#)  
*add*
  - [Share your app](#)
  - [Manage your app](#)
- [Trust and Security](#)
- [Release notes](#)*open in new*
- [Troubleshooting](#)

- [school](#)

### [Knowledge base](#)

- [Tutorials](#)  
*add*
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home/](#)
- [Streamlit Cloud/](#)
- [Get started/](#)
- [Share your app/](#)
- [Configuring Single Sign on \(SSO\)](#)

## **Configuring Single Sign-on (SSO)**

8

By default all apps deployed from private source code are private to the developers in the workspace. Your apps will not be visible to anyone else unless you grant them explicit permission.

If your organization uses **Google OAuth**, you're all set — just click "Continue with Google" to sign into Streamlit Cloud. If your organization uses another SSO provider, check out our guides below.

- [Microsoft Active Directory \(ADFS\)](#)

- [Microsoft Azure authentication](#)
- [Okta](#)
- [Auth0](#)
- [Generic SAML](#)

*push\_pin*

## Note

Questions about SSO? Email us at [success@streamlit.io](mailto:success@streamlit.io)

Was this page helpful?

Yes    No

[edit](#)[Suggest edits](#)

[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

←[Previous: Share your app](#)[Next: Microsoft Active Directory \(ADFS\)](#)→

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [\*cloud\*](#)

### [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notes](#)*open in new*
- [Troubleshooting](#)

- [\*school\*](#)

### [Knowledge base](#)

- [Tutorials](#)  
*remove*
  - [Connect to data sources](#)  
*remove*
    - [AWS S3](#)
    - [BigQuery](#)
    - [Snowflake](#)
    - [Microsoft SQL Server](#)
    - [Firestore](#)*open in new*
    - [MongoDB](#)
    - [MySQL](#)
    - [PostgreSQL](#)
    - [Tableau](#)
    - [Private Google Sheet](#)
    - [Public Google Sheet](#)
    - [TigerGraph](#)
    - [Deta Base](#)
    - [Supabase](#)
    - [Google Cloud Storage](#)
    - [Session State basics](#)
  - [Using Streamlit](#)
  - [Streamlit Components](#)
  - [Installing dependencies](#)
  - [Deployment issues](#)

- [Home/](#)
- [Knowledge base/](#)
- [Tutorials/](#)
- [Connect to data sources/](#)
- [Snowflake](#)

## [Connect Streamlit to Snowflake](#)

*8*

## **Introduction**

*8*

This guide explains how to securely access a Snowflake database from Streamlit Cloud. It uses the [snowflake-connector-python](#) library and Streamlit's [secrets management](#).

## **Create a Snowflake database**

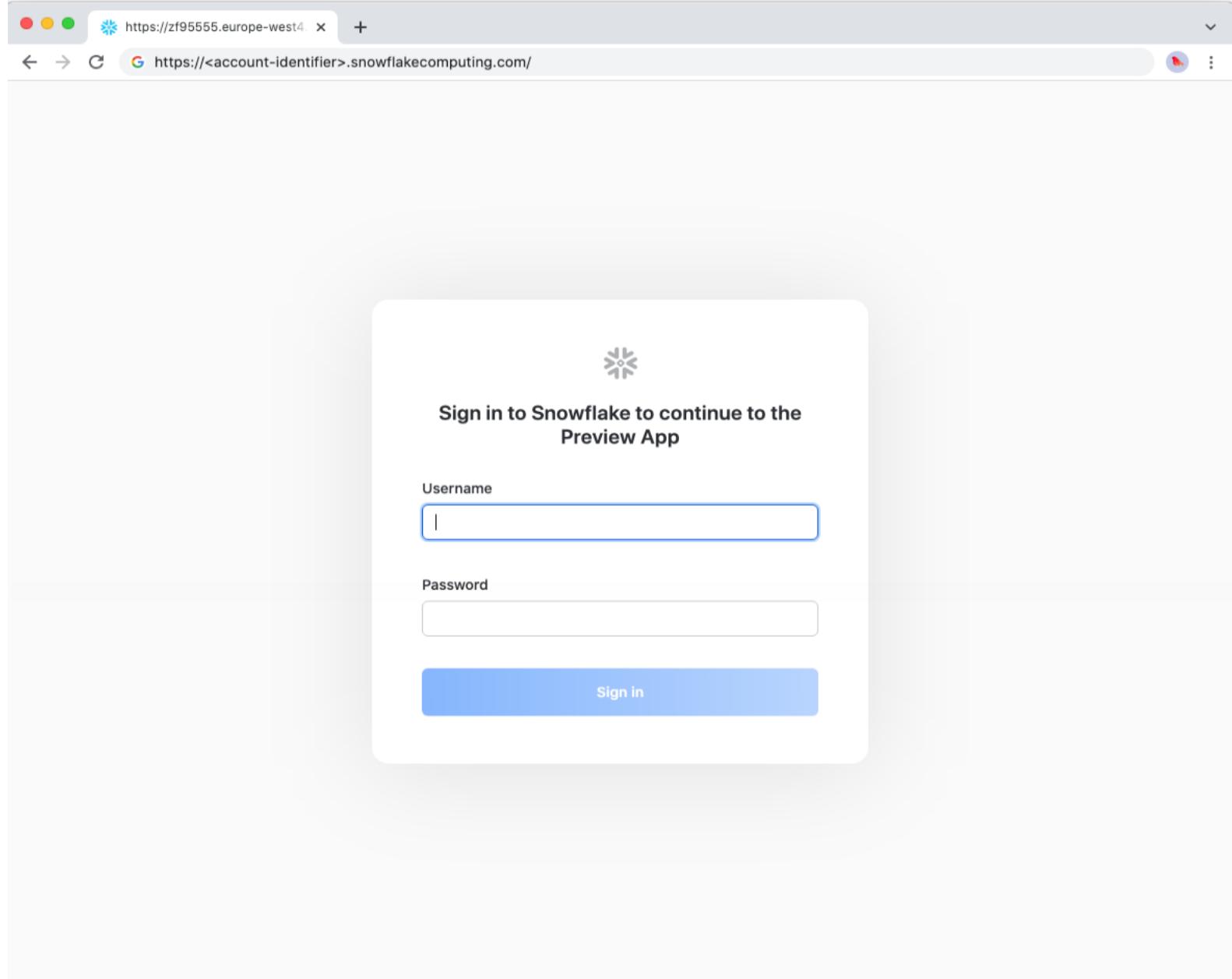
*8*

*push\_pin*

### **Note**

If you already have a database that you want to use, feel free to [skip to the next step](#).

First, [sign up for Snowflake](#) and log into the [Snowflake web interface](#) (note down your username, password, and [account identifier!](#)):



Enter the following queries into the SQL editor in the Worksheets page to create a database and a table with some example values:

```
CREATE DATABASE PETS;  
  
CREATE TABLE MYTABLE (  
    NAME          varchar(80),  
    PET           varchar(80)  
);  
  
INSERT INTO MYTABLE VALUES ('Mary', 'dog'), ('John', 'cat'), ('Robert', 'bird');  
  
SELECT * FROM MYTABLE;
```

Before you execute the queries, first determine which Snowflake UI / web interface you're using. You can either use the [classic web interface](#) or the [new web interface](#).

## Using the Classic Web Interface

*(link)*

To execute the queries in the classic web interface, select **All Queries** and click on **Run**.

The screenshot shows the Snowflake Worksheet interface. At the top, there are navigation icons for Databases, Shares, Data Marketplace, Warehouses, Worksheets (selected), History, Account, Partner Connect, Help, Notifications, and Snowsight. The account name 'SNEHA' is displayed on the right.

In the main area, a 'New Worksheet' tab is selected. A search bar at the top left says 'Find database objects Starting with...'. Below it, two database objects are listed: 'SNOWFLAKE' and 'SNOWFLAKE\_SAMPLE\_DATA'.

The central workspace contains a code editor with the following SQL queries:

```
1 CREATE DATABASE PETS;
2
3 CREATE TABLE MYTABLE (
4     NAME          varchar(80),
5     PET           varchar(80)
6 );
7
8 INSERT INTO MYTABLE VALUES ('Mary', 'dog'), ('John', 'cat'), ('Robert', 'bird');
9
10 SELECT * FROM MYTABLE;
```

At the top of the code editor, there are buttons for 'Run All Queries' (Shift + Cmd/Ctrl + Return) and 'Run (4)' (with a checked 'All Queries' checkbox). The 'Run (4)' button is highlighted with a red box and a cursor icon.

Below the code editor, there are tabs for 'Results' (selected) and 'Data Preview'. A message 'Query results will appear here.' is displayed in the results area.

Make sure to note down the name of your warehouse, database, and schema from the **Context** dropdown menu on the same page:



Databases



Shares



Data Marketplace



Warehouses



Worksheets



History



Account



Partner Connect



Help



Notifications



Snowsight

New Worksheet



Run (4)

 All Queries

Saved 11 minutes ago

Set the Role, Warehouse, Database and Schema for this worksheet. SQL statements referencing objects outside the role context will fail. Setting contexts allows you to reference objects.

Context

Role ACCOUNTADMIN [Change](#)Warehouse COMPUTE\_WH (XS) [On](#)X-Small [Resize](#)

Database PETS

Schema PUBLIC

```

1 CREATE DATABASE PETS;
2
3 CREATE TABLE MYTABLE (
4     NAME          varchar(80),
5     PET           varchar(80)
6 );
7
8 INSERT INTO MYTABLE VALUES ('Mary', 'dog'), ('John'
9
10 SELECT * FROM MYTABLE;

```

Results Data Preview

Open History

 [Query ID](#) [SQL](#) 522ms 3 rows

Filter result...



Copy

Columns

Row	NAME	PET
1	Mary	dog
2	John	cat
3	Robert	bird

## Using the New Web Interface



To execute the queries in the new web interface, highlight or select all the queries with your mouse, and click the play button in the top right corner.

2022-03-03 7:15pm - Snowflake

app.snowflake.com/europe-west4.gcp/zf95555/wwg1wYixfYw#query

2022-03-03 7:15pm

Pinned (0)

No pinned objects

Search

Snowflake

Snowflake\_Sample\_Data

No Database selected

```
CREATE DATABASE PETS;
CREATE TABLE MYTABLE (
    NAME          varchar(80),
    PET           varchar(80)
);
INSERT INTO MYTABLE VALUES ('Mary', 'dog'), ('John', 'cat'), ('Robert', 'bird');
SELECT * FROM MYTABLE;
```

Objects Query

Draft

The screenshot shows the Snowflake web interface. On the left, there's a sidebar with pinned objects, search, and database/schema dropdowns. The main area is a query editor with a blue background. It contains a multi-line SQL script with line numbers from 1 to 10. The first few lines create a database and a table named 'MYTABLE' with columns 'NAME' and 'PET'. Lines 8 and 9 insert three rows of data. Line 10 is a select statement. Below the code, there are tabs for 'Objects' and 'Query', with 'Query' being active. At the top right, there are buttons for 'ACCOUNTADMIN', 'COMPUTE\_WH', 'Share', and a play button with a hand cursor icon. A red box highlights the play button.

priority\_high

## Important

Be sure to highlight or select **all** the queries (lines 1-10) before clicking the play button.

Once you have executed the queries, you should see a preview of the table in the **Results** panel at the bottom of the page. Additionally, you should see your newly created database and schema by expanding the accordion on the left side of the page. Lastly, the warehouse name is displayed on the button to the left of the **Share** button.

The screenshot shows the Snowflake UI with the following details:

- Top Bar:** Shows the date and time (2022-03-03 2:34pm) and the URL (app.snowflake.com/europe-west4.gcp/zf95555/w4cnqlnCINEX#query).
- Left Sidebar:** Includes sections for "Pinned (0)", "No pinned objects", and a "Search" bar. The "PETS" and "PUBLIC" schemas are listed under "Objects" and are highlighted with red boxes.
- Middle Panel:** Shows two dropdown menus: "Roles" and "Warehouses". The "ACCOUNTADMIN" role is selected in the Roles menu. The "COMPUTE\_WH" warehouse is selected in the Warehouses menu and is also highlighted with a red box.
- Bottom Panel:** Displays a table titled "Objects" with three rows:

	NAME	PET
1	Mary	dog
2	John	cat
3	Robert	bird

A "Query Details" sidebar on the right shows "Query duration: 599ms" and "Rows: 3".

Make sure to note down the name of your warehouse, database, and schema.

## Add username and password to your local app secrets

Your local Streamlit app will read secrets from a file `.streamlit/secrets.toml` in your app's root directory. Create this file if it doesn't exist yet and add your Snowflake username, password, [account identifier](#), and the name of your warehouse, database, and schema as shown below:

```
# .streamlit/secrets.toml

[snowflake]
user = "xxxx"
password = "xxxx"
account = "xxxx"
warehouse = "xxxx"
database = "xxx"
schema = "xxx"
```

If you created the database from the previous step, the names of your database and schema are `PETS` and `PUBLIC`, respectively.

`priority_high`

### Important

Add this file to `.gitignore` and don't commit it to your Github repo!

## Copy your app secrets to the cloud

As the `secrets.toml` file above is not committed to Github, you need to pass its content to your deployed app (on Streamlit Cloud) separately. Go to the [app dashboard](#) and in the app's dropdown menu, click on [Edit Secrets](#). Copy the content of `secrets.toml` into the text area. More information is available at [Secrets Management](#).

The screenshot shows a Streamlit app settings page. On the left, there's a sidebar with 'App settings', 'Sharing', and 'Secrets' (which is currently selected). The main area is titled 'Secrets' and contains instructions about providing environment variables and secrets using TOML format. It shows a code editor with the following TOML configuration:

```
DB_USERNAME = "myuser"  
DB_TOKEN = "abcdef"  
  
[some_section]  
some_key = 1234
```

A blue 'Save' button is located at the bottom right of the editor.

## Add snowflake-connector-python to your requirements file

8

Add the [snowflake-connector-python](#) package to your `requirements.txt` file, preferably pinning its version (replace `x.x.x` with the version you want installed):

```
# requirements.txt  
snowflake-connector-python==x.x.x
```

## Write your Streamlit app

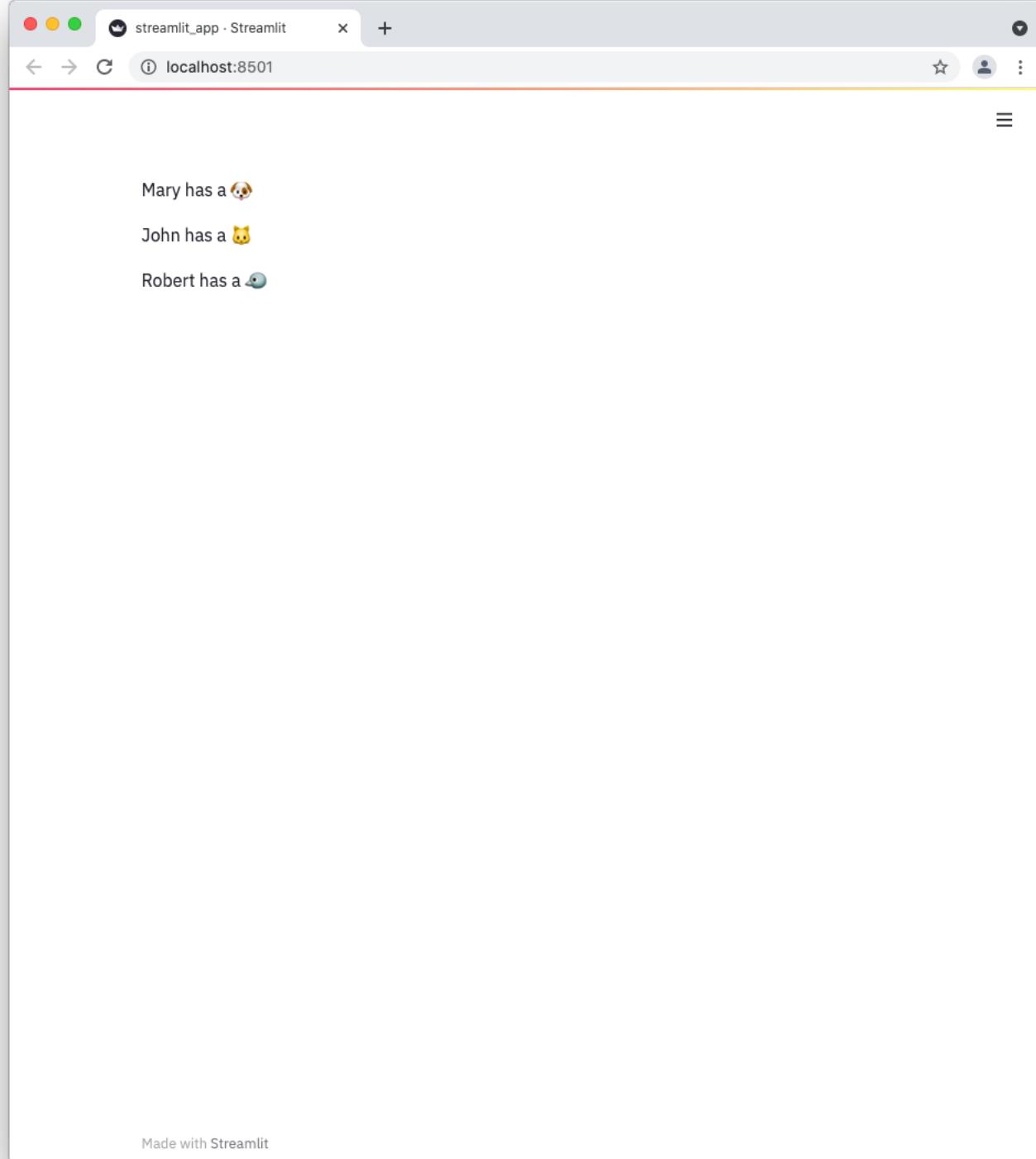
8

Copy the code below to your Streamlit app and run it. Make sure to adapt query to use the name of your table.

```
# streamlit_app.py  
  
import streamlit as st  
import snowflake.connector  
  
# Initialize connection.  
# Uses st.experimental_singleton to only run once.  
@st.experimental_singleton  
def init_connection():  
    return snowflake.connector.connect(**st.secrets["snowflake"])  
  
conn = init_connection()  
  
# Perform query.  
# Uses st.experimental_memo to only rerun when the query changes or after 10 min.  
@st.experimental_memo(ttl=600)  
def run_query(query):  
    with conn.cursor() as cur:  
        cur.execute(query)  
        return cur.fetchall()  
  
rows = run_query("SELECT * from mytable;")  
  
# Print results.  
for row in rows:  
    st.write(f"{row[0]} has a :{row[1]}:")
```

See `st.experimental_memo` above? Without it, Streamlit would run the query every time the app reruns (e.g. on a widget interaction). With `st.experimental_memo`, it only runs when the query changes or after 10 minutes (that's what `ttl` is for). Watch out: If your database updates more frequently, you should adapt `ttl` or remove caching so viewers always see the latest data. Read more about caching [here](#).

If everything worked out (and you used the example table we created above), your app should look like this:



Was this page helpful?

Yes  No  
[edit](#)[Suggest edits](#)  
[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

[← Previous: BigQuery](#) [Next: Microsoft SQL Server →](#)

---

[Home](#) [Contact Us](#) [Community](#)



# st.altair\_chart

v1.9.0

Display a chart using the Altair library.

## Function signature

```
st.altair_chart(altair_chart, use_container_width=False)
```

## Parameters

**altair\_chart** (*altair.vegalite.v2.api.Chart*)

The Altair chart object to display.

**use\_container\_width** (*bool*)

If True, set the chart width to the column width. This takes precedence over Altair's native *width* value.

## Example

```
import pandas as pd
import numpy as np
import altair as alt

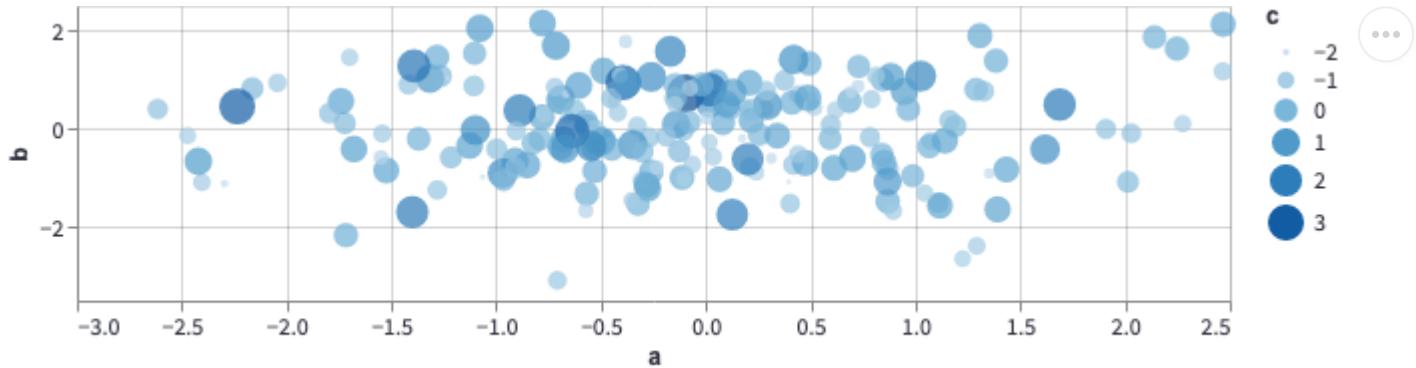
df = pd.DataFrame(
    np.random.randn(200, 3),
    columns=['a', 'b', 'c'])

c = alt.Chart(df).mark_circle().encode(
    x='a', y='b', size='c', color='c', tooltip=['a', 'b', 'c'])
```

```
st.altair_chart(c, use_container_width=True)
```



Examples of Altair charts can be found at <https://altair-viz.github.io/gallery/>.



([view standalone Streamlit app](#)).

## Annotating charts

Altair also allows you to annotate your charts with text, images, and emojis. You can do this by creating [layered charts](#), which let you overlay two different charts on top of each other. The idea is to use the first chart to show the data, and the second chart to show the annotations. The second chart can then be overlaid on top of the first chart using the operator to create a layered chart.

Let's walk through an example of annotating a time-series chart with text and an emoji.

### Step 1: Create the base chart

In this example, we create a time-series chart to track the evolution of stock prices. The chart is interactive and contains a multi-line tooltip. Click [here](#) to learn more about multi-line tooltips in Altair.

First, we import the required libraries and load the example stocks dataset using the [vega\\_datasets](#) package:

```
import altair as alt
import pandas as pd
import streamlit as st
from vega_datasets import data

# We use @st.experimental_memo to keep the dataset in cache
@st.experimental_memo
def get_data():
```



```
source = data.stocks()
source = source[source.date.gt("2004-01-01")]
return source

source = get_data()
```

Next, we define a function `get_chart()` to create the interactive time-series chart of the stock prices with a multi-line tooltip. The x-axis represents the date, and the y-axis represents the stock price.

We then invoke `get_chart()` that takes the stock prices dataframe as an input and returns a chart object. This is going to be our base chart on which we will overlay the annotations in Step 2.

```
# Define the base time-series chart.
def get_chart(data):
    hover = alt.selection_single(
        fields=["date"],
        nearest=True,
        on="mouseover",
        empty="none",
    )

    lines = (
        alt.Chart(data, title="Evolution of stock prices")
        .mark_line()
        .encode(
            x="date",
            y="price",
            color="symbol",
        )
    )

    # Draw points on the line, and highlight based on selection
    points = lines.transform_filter(hover).mark_circle(size=65)

    # Draw a rule at the location of the selection
    tooltips = (
        alt.Chart(data)
        .mark_rule()
        .encode(
            x="yearmonthdate(date)",
            y="price",
```

```
        opacity=alt.condition(hover, alt.value(0.3), alt.value(0)),
        tooltip=[
            alt.Tooltip("date", title="Date"),
            alt.Tooltip("price", title="Price (USD)"),
        ],
    )
    .add_selection(hover)
)
return (lines + points + tooltips).interactive()

chart = get_chart(source)
```

## Step 2: Annotate the chart

Now that we have our first chart that shows the data, we can annotate it with text and an emoji. Let's overlay the ↓ emoji on top of the time-series chart at specific points of interest. We want users to hover over the ↓ emoji to see the associated annotation text.

For simplicity, let's annotate four specific dates and set the height of the annotations at constant value of 10.

### ★ Tip

You can vary the horizontal and vertical positions of the annotations by replacing the hard-coded values with the output of Streamlit widgets! Click [here](#) to jump to a live example below, and develop an intuition for the ideal horizontal and vertical positions of the annotations by playing with Streamlit widgets.

To do so, we create a dataframe annotations\_df containing the dates, annotation text, and the height of the annotations:

```
# Add annotations
ANNOTATIONS = [
    ("Mar 01, 2008", "Pretty good day for GOOG"),
    ("Dec 01, 2007", "Something's going wrong for GOOG & AAPL"),
    ("Nov 01, 2008", "Market starts again thanks to..."),
    ("Dec 01, 2009", "Small crash for GOOG after..."),
]
annotations_df = pd.DataFrame(ANNOTATIONS, columns=["date", "event"])
annotations_df.date = pd.to_datetime(annotations_df.date)
annotations_df["y"] = 10
```

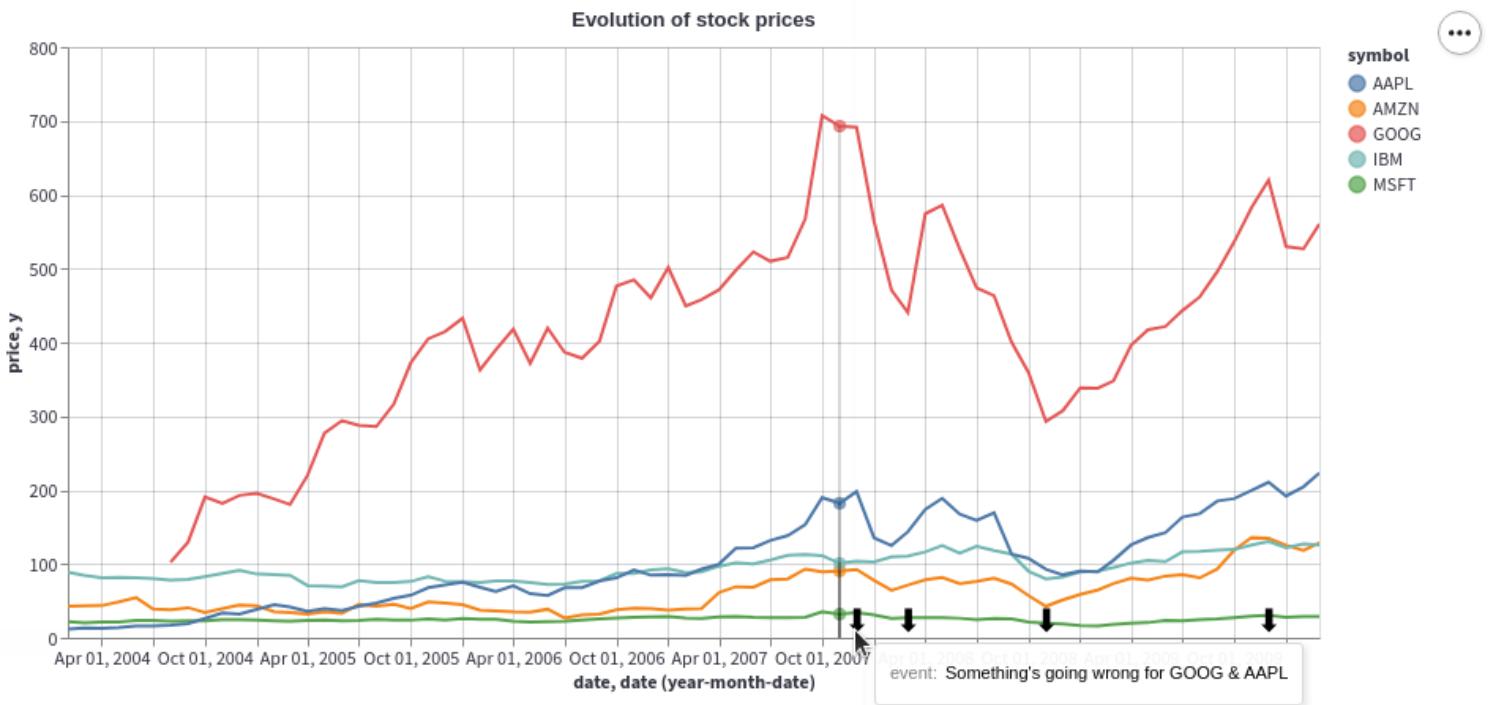
Using this dataframe, we create a scatter plot with the x-axis representing the date, and the y-axis representing the height of the annotation. The data point at the specific date and height is represented by the ↓ emoji, using Altair's `mark_text()` `mark`.

The annotation text is displayed as a tooltip when users hover over the ↓ emoji. This is achieved using Altair's `encode()` method to map the `event` column containing the annotation text to the visual attribute ↓ of the plot.

```
annotation_layer = (
    alt.Chart(annotations_df)
    .mark_text(size=20, text="⬇", dx=-8, dy=-10, align="left")
    .encode(
        x="date:T",
        y=alt.Y("y:Q"),
        tooltip=["event"],
    )
    .interactive()
)
```

Finally, we overlay the annotations on top of the base chart using the + operator to create the final layered chart! 🎉

```
st.altair_chart(
    (chart + annotation_layer).interactive(),
    use_container_width=True
)
```



To use images instead of emojis, replace the line containing `.mark_text()` with `.mark_image()`, and replace `image_url` below with the URL of the image:

```
.mark_image(
    width=12,
    height=12,
    url="image_url",
)
```

## Interactive example

Now that you've learned how to annotate charts, the sky's the limit! We've extended the above example to let you interactively paste your favorite emoji and set its position on the chart with Streamlit widgets.



(view standalone Streamlit app)

## Was this page helpful?

 Yes

 No

 [Suggest edits](#)



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

[\*\*← Previous:\*\* st.pyplot](#)

**Next:** st.vega\_lite\_chart →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# st.area\_chart

v1.9.0

Display an area chart.

This is just syntax-sugar around st.altair\_chart. The main difference is this command uses the data's own column and indices to figure out the chart's spec. As a result this is easier to use for many "just plot this" scenarios, while being less customizable.

If st.area\_chart does not guess the data specification correctly, try specifying your desired chart using st.altair\_chart.

## Function signature

```
st.area_chart(data=None, width=0, height=0, use_container_width=True)
```

## Parameters

**data** (*pandas.DataFrame*, *pandas.Styler*, *pyarrow.Table*, *numpy.ndarray*, *Iterable*, or *dict*)

Data to be plotted. Pyarrow tables are not supported by Streamlit's legacy DataFrame serialization (i.e. with *config.dataFrameSerialization = "legacy"*). To use pyarrow tables, please enable pyarrow by changing the config setting, *config.dataFrameSerialization = "arrow"*.

### width (*int*)

The chart width in pixels. If 0, selects the width automatically.

### height (*int*)

The chart height in pixels. If 0, selects the height automatically.

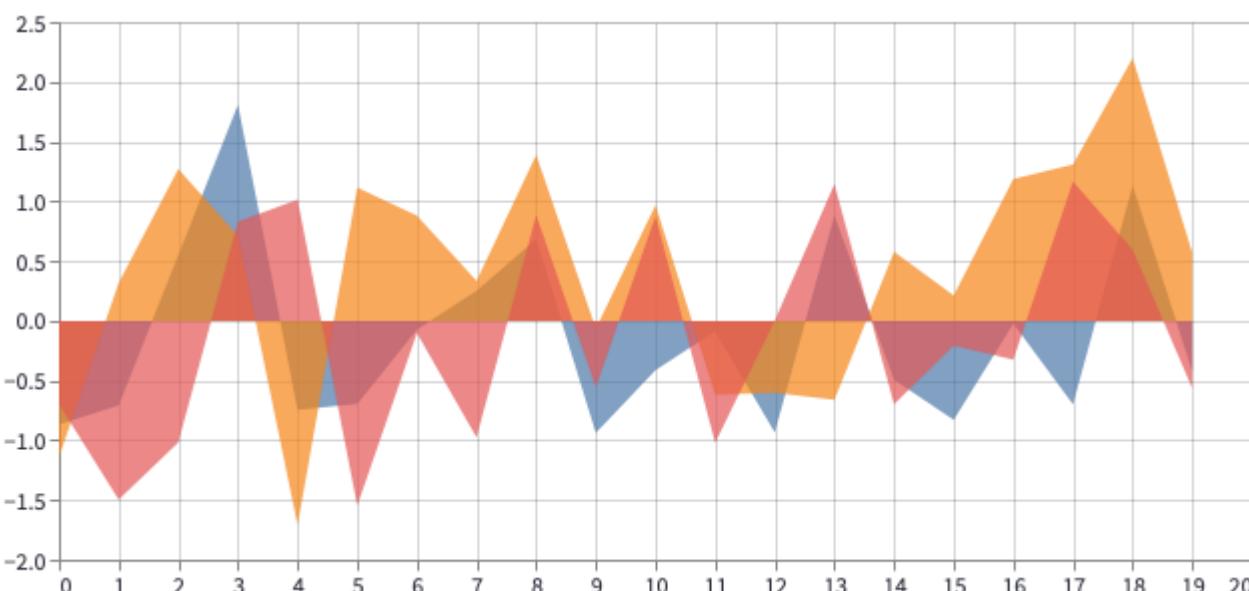
## `use_container_width (bool)`

If True, set the chart width to the column width. This takes precedence over the width argument.

### Example

```
chart_data = pd.DataFrame(  
    np.random.randn(20, 3),  
    columns=['a', 'b', 'c'])
```

```
st.area_chart(chart_data)
```



(view standalone Streamlit app)

### Was this page helpful?

Yes    No

[Suggest edits](#)



### Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

[\*\*← Previous:\*\* st.line\\_chart](#)

**Next:** st.bar\_chart →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.

## Documentation

*search*

Search

- *description*

### Streamlit library

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*add*
  - [Media elements](#)  
*remove*
    - [st.image](#)
    - [st.audio](#)
    - [st.video](#)
  - [Layouts and containers](#)  
*add*
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- *cloud*

### Streamlit Cloud

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notes](#)*open in new*
- [Troubleshooting](#)

- *school*

### Knowledge base

- [Tutorials](#)
- [add](#)
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home/](#)
- [Streamlit library/](#)
- [API reference/](#)
- [Media elements/](#)
- [st.audio](#)

## st.audio

8

Streamlit Version v1.9.0 ▾

Display an audio player.

### Function signature

**st.audio(data, format="audio/wav", start\_time=0)**

#### Parameters

data (str, bytes, BytesIO, io.open()). Raw audio data, filename, or a URL pointing to the file to load. Numpy arrays numpy.ndarray, or file opened and raw data formats must include all necessary file headers to match specified file with)

start\_time (int) The time from which this element should start playing.

format (str) The mime type for the audio file. Defaults to 'audio/wav'. See <https://tools.ietf.org/html/rfc4281> for more info.

#### Example

```
audio_file = open('myaudio.ogg', 'rb')
audio_bytes = audio_file.read()

st.audio(audio_bytes, format='audio/ogg')
```

Please wait...

Made with Streamlit

([view standalone Streamlit app](#)).

Was this page helpful?

Yes No

[edit](#)[Suggest edits](#)  
[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

←[Previous: st.image](#)[Next: st.video](#)→

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.



# st.balloons

v1.9.0

Draw celebratory balloons.

## Function signature

```
st.balloons()
```

## Example

```
st.balloons()
```

...then watch your app and get ready for a celebration!

## Was this page helpful?

Yes

No

[Suggest edits](#)



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# st.bar\_chart

v1.9.0

Display a bar chart.

This is just syntax-sugar around st.altair\_chart. The main difference is this command uses the data's own column and indices to figure out the chart's spec. As a result this is easier to use for many "just plot this" scenarios, while being less customizable.

If st.bar\_chart does not guess the data specification correctly, try specifying your desired chart using st.altair\_chart.

## Function signature

```
st.bar_chart(data=None, width=0, height=0, use_container_width=True)
```

## Parameters

**data** (*pandas.DataFrame*, *pandas.Styler*, *pyarrow.Table*, *numpy.ndarray*, *Iterable*, or *dict*)

Data to be plotted. Pyarrow tables are not supported by Streamlit's legacy DataFrame serialization (i.e. with *config.dataFrameSerialization = "legacy"*). To use pyarrow tables, please enable pyarrow by changing the config setting, *config.dataFrameSerialization = "arrow"*.

### width (*int*)

The chart width in pixels. If 0, selects the width automatically.

### height (*int*)

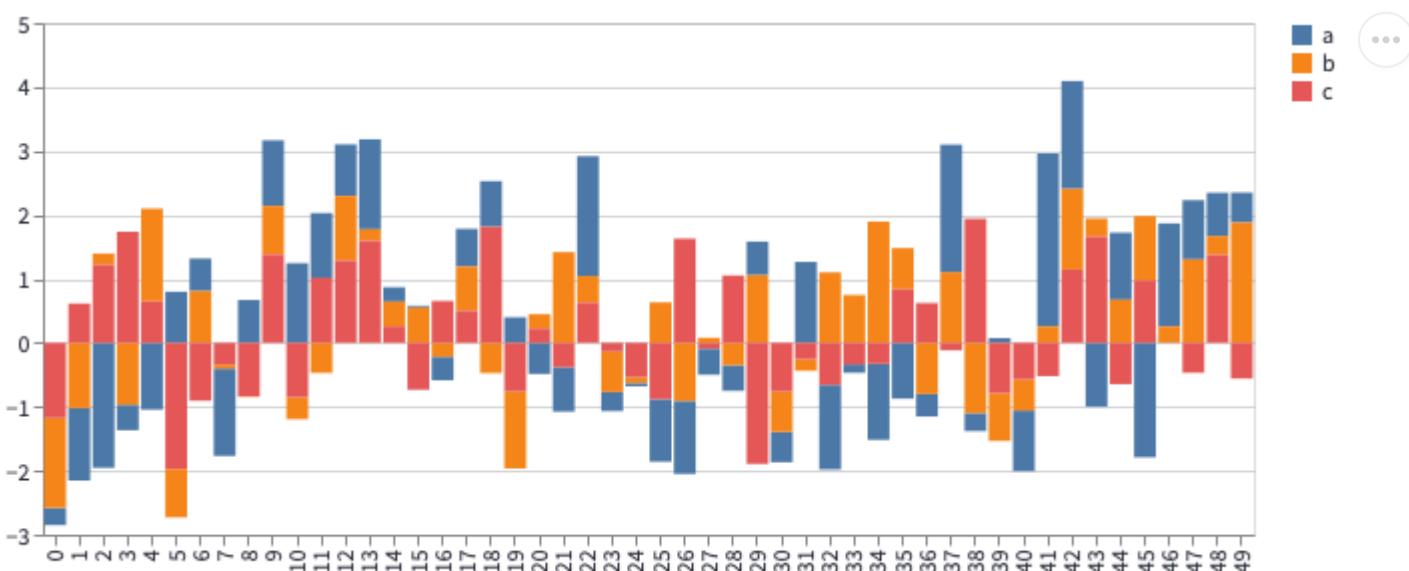
The chart height in pixels. If 0, selects the height automatically.

## use\_container\_width (bool)

If True, set the chart width to the column width. This takes precedence over the width argument.

## Example

```
chart_data = pd.DataFrame(  
    np.random.randn(50, 3),  
    columns=["a", "b", "c"])  
  
st.bar_chart(chart_data)
```



[\(view standalone Streamlit app\)](#)

## Was this page helpful?

Yes

No

Suggest edits



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

← [Previous: st.area\\_chart](#)

[Next: st.pyplot](#) →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# st.bokeh\_chart

v1.9.0

Display an interactive Bokeh chart.

Bokeh is a charting library for Python. The arguments to this function closely follow the ones for Bokeh's `show` function. You can find more about Bokeh at <https://bokeh.pydata.org>.

To show Bokeh charts in Streamlit, call `st.bokeh_chart` wherever you would call Bokeh's `show`.

## Function signature

```
st.bokeh_chart(figure, use_container_width=False)
```

## Parameters

**figure** (`bokeh.plotting.figure.Figure`)

A Bokeh figure to plot.

**use\_container\_width** (`bool`)

If True, set the chart width to the column width. This takes precedence over Bokeh's native `width` value.

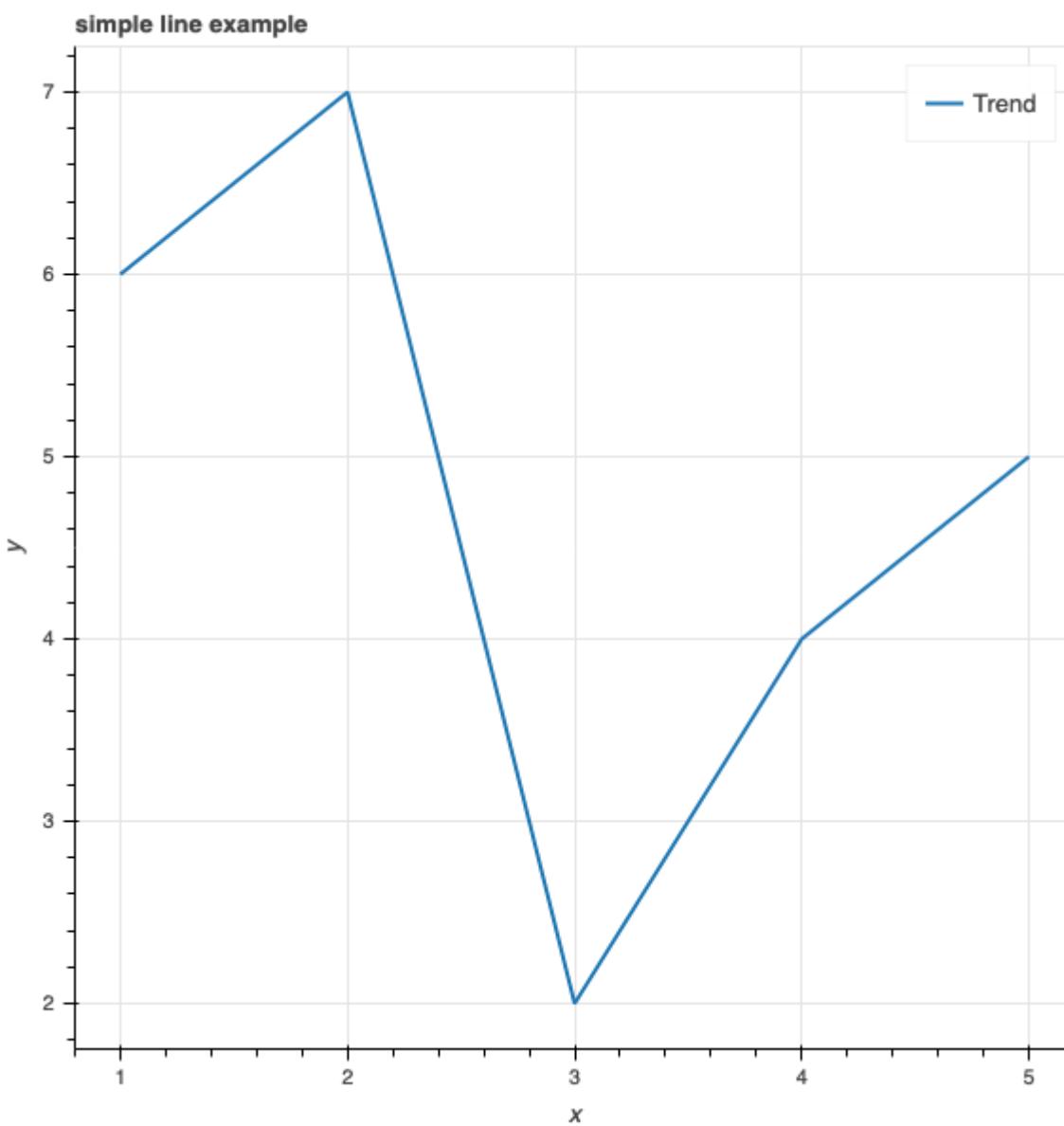
## Example

```
import streamlit as st
from bokeh.plotting import figure

x = [1, 2, 3, 4, 5]
y = [6, 7, 2, 4, 5]
```



```
p = figure(  
    title='simple line example',  
    x_axis_label='x',  
    y_axis_label='y')  
  
p.line(x, y, legend_label='Trend', line_width=2)  
  
st.bokeh_chart(p, use_container_width=True)
```



[\(view standalone Streamlit app\)](#)

Was this page helpful?

Yes    No

 [Suggest edits](#)

---

## Still have questions?

 Our [forums](#) are full of helpful information and Streamlit experts.

---

← [Previous: st.plotly\\_chart](#)

[Next: st.pydeck\\_chart](#) →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*remove*
    - [st.button](#)
    - [st.download\\_button](#)
    - [st.checkbox](#)
    - [st.radio](#)
    - [st.selectbox](#)
    - [st.multiselect](#)
    - [st.slider](#)
    - [st.select\\_slider](#)
    - [st.text\\_input](#)
    - [st.number\\_input](#)
    - [st.text\\_area](#)
    - [st.date\\_input](#)
    - [st.time\\_input](#)
    - [st.file\\_uploader](#)
    - [st.camera\\_input](#)
    - [st.color\\_picker](#)
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*add*
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)

- [Cheat sheet](#)

- [cloud](#)

## [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notes](#)*open in new*
- [Troubleshooting](#)

- [school](#)

## [Knowledge base](#)

- [Tutorials](#)  
*add*
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home](#)/
- [Streamlit library](#)/
- [API reference](#)/
- [Input widgets](#)/
- [st.button](#)

## st.button

8

Streamlit Version [v1.9.0](#) ▾

Display a button widget.

### Function signature

```
st.button(label, key=None, help=None, on_click=None, args=None, kwargs=None, *, disabled=False)
```

#### Parameters

**label** (str) A short label explaining to the user what this button is for.

**key** (str or int) An optional string or integer to use as the unique key for the widget. If this is omitted, a key will be generated for the widget based on its content. Multiple widgets of the same type may not share the same key.

**help** (str) An optional tooltip that gets displayed when the button is hovered over.

**on\_click** (callable) An optional callback invoked when this button is clicked.

**args** (tuple) An optional tuple of args to pass to the callback.

**kwargs** (dict) An optional dict of kwargs to pass to the callback.

**disabled** An optional boolean, which disables the button if set to True. The default is False. This argument can only be

(bool) supplied by keyword.

## Returns

(bool) True if the button was clicked on the last run of the app, False otherwise.

## Example

```
if st.button('Say hello'):
    st.write('Why hello there')
else:
    st.write('Goodbye')
```

[\(view standalone Streamlit app\)](#)

## Featured videos



Check out our video on how to use one of Streamlit's core functions, the button!

Streamlit Shorts: Ho...



In the video below, we'll take it a step further and learn how to combine a [button](#), [checkbox](#) and [radio button](#)!

Streamlit Shorts: Ho...



Was this page helpful?

Yes  No

[edit](#) [Suggest edits](#)

[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: Input widgets](#) [Next: st.download\\_button](#) →

---

[Home](#) [Contact Us](#) [Community](#)



Copyright © 2022, Streamlit Inc.



# st.cache

When you mark a function with Streamlit's cache annotation, it tells Streamlit that whenever the function is called it should check three things:

1. The name of the function
2. The actual code that makes up the body of the function
3. The input parameters that you called the function with

If this is the first time Streamlit has seen those three items, with those exact values, and in that exact combination, it runs the function and stores the result in a local cache.

Then, next time the function is called, if those three values have not changed Streamlit knows it can skip executing the function altogether. Instead, it just reads the output from the local cache and passes it on to the caller.

The main limitation is that Streamlit's cache feature doesn't know about changes that take place outside the body of the annotated function.

For more information about the Streamlit cache, its configuration parameters, and its limitations, see [Caching](#).

# st.cache

## v1.9.0

Function decorator to memoize function executions.

### Function signature

```
st.cache(func=None, persist=False, allow_output_mutation=False,  
show_spinner=True, suppress_st_warning=False, hash_funcs=None, max_entries=None,  
ttl=None)
```

### Parameters

## **func** (*callable*)

The function to cache. Streamlit hashes the function and dependent code.

## **persist** (*boolean*)

Whether to persist the cache on disk.

## **allow\_output\_mutation** (*boolean*)

Streamlit shows a warning when return values are mutated, as that can have unintended consequences. This is done by hashing the return value internally.

If you know what you're doing and would like to override this warning, set this to True.

## **show\_spinner** (*boolean*)

Enable the spinner. Default is True to show a spinner when there is a cache miss.

## **suppress\_st\_warning** (*boolean*)

Suppress warnings about calling Streamlit functions from within the cached function.

## **hash\_funcs** (*dict or None*)

Mapping of types or fully qualified names to hash functions. This is used to override the behavior of the hasher inside Streamlit's caching mechanism: when the hasher encounters an object, it will first check to see if its type matches a key in this dict and, if so, will use the provided function to generate a hash for it. See below for an example of how this can be used.

## **max\_entries** (*int or None*)

The maximum number of entries to keep in the cache, or None for an unbounded cache. (When a new entry is added to a full cache, the oldest cached entry will be removed.) The default is None.

## **ttl** (*float or None*)

The maximum number of seconds to keep an entry in the cache, or None if cache entries should not expire. The default is None.

## Example

```
@st.cache
def fetch_and_clean_data(url):
    # Fetch data from URL here, and then clean it up.
    return data

d1 = fetch_and_clean_data(DATA_URL_1)
# Actually executes the function, since this is the first time it was
# encountered.

d2 = fetch_and_clean_data(DATA_URL_1)
# Does not execute the function. Instead, returns its previously computed
# value. This means that now the data in d1 is the same as in d2.

d3 = fetch_and_clean_data(DATA_URL_2)
# This is a different URL, so the function executes.
```

To set the `persist` parameter, use this command as follows:

```
@st.cache(persist=True)
def fetch_and_clean_data(url):
    # Fetch data from URL here, and then clean it up.
    return data
```

To disable hashing return values, set the `allow_output_mutation` parameter to `True`:

```
@st.cache(allow_output_mutation=True)
def fetch_and_clean_data(url):
    # Fetch data from URL here, and then clean it up.
    return data
```

To override the default hashing behavior, pass a custom hash function. You can do that by mapping a type (e.g. `MongoClient`) to a hash function (`__hash__`) like this:

```
@st.cache(hash_funcs={MongoClient: id})  
def connect_to_database(url):  
    return MongoClient(url)
```



Alternatively, you can map the type's fully-qualified name (e.g. "pymongo.mongo\_client.MongoClient") to the hash function instead:

```
@st.cache(hash_funcs={"pymongo.mongo_client.MongoClient": id})  
def connect_to_database(url):  
    return MongoClient(url)
```



## Was this page helpful?

Yes    No

[Suggest edits](#)



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

[← Previous: Performance](#)

[Next: st.experimental\\_memo →](#)





# st.caption

v1.9.0

Display text in small font.

This should be used for captions, asides, footnotes, sidenotes, and other explanatory text.

## Function signature

```
st.caption(body, unsafe_allow_html=False)
```

## Parameters

### body (str)

The text to display.

### unsafe\_allow\_html (bool)

By default, any HTML tags found in strings will be escaped and therefore treated as pure text. This behavior may be turned off by setting this argument to True.

That said, we *strongly advise against it*. It is hard to write secure HTML, so by using this argument you may be compromising your users' security. For more information, see:

<https://github.com/streamlit/streamlit/issues/152>

**Also note that `unsafe\_allow\_html` is a temporary measure and may be removed from Streamlit at any time.**

If you decide to turn on HTML anyway, we ask you to please tell us your exact use case here:

<https://discuss.streamlit.io/t/96> .

This will help us come up with safe APIs that allow you to do what you want.

## Example

```
st.caption('This is a string that explains something above.')
```

This is a caption

## Was this page helpful?

 Yes     No

 [Suggest edits](#)

---

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[← Previous: st.subheader](#)

[Next: st.code →](#)

---

[Home](#)

[Contact Us](#)

[Community](#)



Copyright © 2022, Streamlit Inc.



## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*remove*
    - [st.button](#)
    - [st.download\\_button](#)
    - [st.checkbox](#)
    - [st.radio](#)
    - [st.selectbox](#)
    - [st.multiselect](#)
    - [st.slider](#)
    - [st.select\\_slider](#)
    - [st.text\\_input](#)
    - [st.number\\_input](#)
    - [st.text\\_area](#)
    - [st.date\\_input](#)
    - [st.time\\_input](#)
    - [st.file\\_uploader](#)
    - [st.camera\\_input](#)
    - [st.color\\_picker](#)
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*add*
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)

- [Cheat sheet](#)

- [cloud](#)

## [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notes](#)*open in new*
- [Troubleshooting](#)

- [school](#)

## [Knowledge base](#)

- [Tutorials](#)  
*add*
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home](#)/
- [Streamlit library](#)/
- [API reference](#)/
- [Input widgets](#)/
- [st.checkbox](#)

## st.checkbox

8

Streamlit Version [v1.9.0](#) 

Display a checkbox widget.

### Function signature

```
st.checkbox(label, value=False, key=None, help=None, on_change=None, args=None, kwargs=None, *,  
           disabled=False)
```

#### Parameters

**label** (str) A short label explaining to the user what this checkbox is for.

**value** (bool) Preselect the checkbox when it first renders. This will be cast to bool internally.

**key** (str or int) An optional string or integer to use as the unique key for the widget. If this is omitted, a key will be generated for the widget based on its content. Multiple widgets of the same type may not share the same key.

**help** (str) An optional tooltip that gets displayed next to the checkbox.

**on\_change** (callable) An optional callback invoked when this checkbox's value changes.

**args** (tuple) An optional tuple of args to pass to the callback.

`kwargs` An optional dict of kwargs to pass to the callback.  
(dict)

`disabled` An optional boolean, which disables the checkbox if set to True. The default is False. This argument can  
(bool) only be supplied by keyword.

Returns

(bool) Whether or not the checkbox is checked.

## Example

```
agree = st.checkbox('I agree')

if agree:
    st.write('Great!')
```



Please wait...

[\(view standalone Streamlit app\)](#)

## Featured videos

8

Check out our video on how to use one of Streamlit's core functions, the checkbox! ☑

Streamlit Shorts: Ho...



In the video below, we'll take it a step further and learn how to combine a [button](#), [checkbox](#) and [radio button](#)!

Streamlit Shorts: Ho...



Was this page helpful?

[thumb\\_upYes](#) [thumb\\_downNo](#)

[edit](#)[Suggest edits](#)

[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

←[Previous: st.download\\_button](#)[Next: st.radio](#)→

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.

## Documentation

*search*

Search

- *description*

### Streamlit library

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*remove*
    - [st.markdown](#)
    - [st.title](#)
    - [st.header](#)
    - [st.subheader](#)
    - [st.caption](#)
    - [st.code](#)
    - [st.text](#)
    - [st.latex](#)
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*add*
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*add*
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- *cloud*

### Streamlit Cloud

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notes](#)*open in new*

- o [Troubleshooting](#)

- [school](#)

## [Knowledge base](#)

- o [Tutorials](#)

- add*

- o [Using Streamlit](#)

- o [Streamlit Components](#)

- o [Installing dependencies](#)

- o [Deployment issues](#)

- [Home](#)/
- [Streamlit library](#)/
- [API reference](#)/
- [Text elements](#)/
- [st.code](#)

## st.code

8

Streamlit Version v1.9.0 ▾

Display a code block with optional syntax highlighting.

(This is a convenience wrapper around `st.markdown()`)

### Function signature

```
st.code(body, language="python")
```

#### Parameters

body (str) The string to display as code.

language (str) The language that the code is written in, for syntax highlighting. If omitted, the code will be unstyled.

#### Example

```
code = '''def hello():
    print("Hello, Streamlit!")'''
st.code(code, language='python')
```

```
def hello():
    print("Hello, Streamlit!")
```



Was this page helpful?

Yes  No

[edit](#)[Suggest edits](#)

[forum](#)

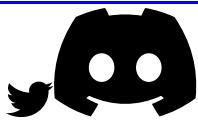
## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: st.caption](#) [Next: st.text](#) →

---

[Home](#) [Contact Us](#) [Community](#)



Copyright © 2022, Streamlit Inc.



# st.color\_picker

v1.9.0

Display a color picker widget.

## Function signature

```
st.color_picker(label, value=None, key=None, help=None, on_change=None,  
args=None, kwargs=None, *, disabled=False)
```

## Parameters

### label (str)

A short label explaining to the user what this input is for.

### value (str)

The hex value of this widget when it first renders. If None, defaults to black.

### key (str or int)

An optional string or integer to use as the unique key for the widget. If this is omitted, a key will be generated for the widget based on its content. Multiple widgets of the same type may not share the same key.

### help (str)

An optional tooltip that gets displayed next to the color picker.

### on\_change (callable)

An optional callback invoked when this color\_picker's value changes.

### **args** (*tuple*)

An optional tuple of args to pass to the callback.

### **kwargs** (*dict*)

An optional dict of kwargs to pass to the callback.

### **disabled** (*bool*)

An optional boolean, which disables the color picker if set to True. The default is False. This argument can only be supplied by keyword.

### **Returns**

#### *(str)*

The selected color as a hex string.

## **Example**

```
color = st.color_picker('Pick A Color', '#00f900')
st.write('The current color is', color)
```



Pick A Color



The current color is #00f900

[\(view standalone Streamlit app\)](#)

Was this page helpful?

Yes

No

 Suggest edits

---

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

← Previous: [st.camera\\_input](#)

Next: [Media elements](#) →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*add*
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*remove*
    - [st.sidebar](#)
    - [st.columns](#)
    - [st.expander](#)
    - [st.container](#)
    - [st.empty](#)
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [\*cloud\*](#)

### [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notesopen in new](#)
- [Troubleshooting](#)

- [\*school\*](#)

## [Knowledge base](#)

- [Tutorials](#)  
*add*
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home/](#)
- [Streamlit library/](#)
- [API reference/](#)
- [Layouts and containers/](#)
- [st.columns](#)

## st.columns

8

Streamlit Version [v1.9.0](#) ▾

Insert containers laid out as side-by-side columns.

Inserts a number of multi-element containers laid out side-by-side and returns a list of container objects.

To add elements to the returned containers, you can use "with" notation (preferred) or just call methods directly on the returned object. See examples below.

### Warning

Currently, you may not put columns inside another column.

#### Function signature

**st.columns(spec)**

#### Parameters

If an int

Specifies the number of columns to insert, and all columns have equal width.

If a list of numbers

spec (int or list  
of numbers)

Creates a column for each number, and each column's width is proportional to the number provided. Numbers can be ints or floats, but they must be positive.

For example, `st.columns([3, 1, 2])` creates 3 columns where the first column is 3 times the width of the second, and the last column is 2 times that width.

#### Returns

(list of  
containers)

A list of container objects.

## Examples

You can use *with* notation to insert any element into a column:

```
col1, col2, col3 = st.columns(3)
```

```
with col1:  
    st.header("A cat")
```

```
st.image("https://static.streamlit.io/examples/cat.jpg")

with col2:
    st.header("A dog")
    st.image("https://static.streamlit.io/examples/dog.jpg")

with col3:
    st.header("An owl")
    st.image("https://static.streamlit.io/examples/owl.jpg")
```



Please wait...

Made with Streamlit

[\(view standalone Streamlit app\)](#)

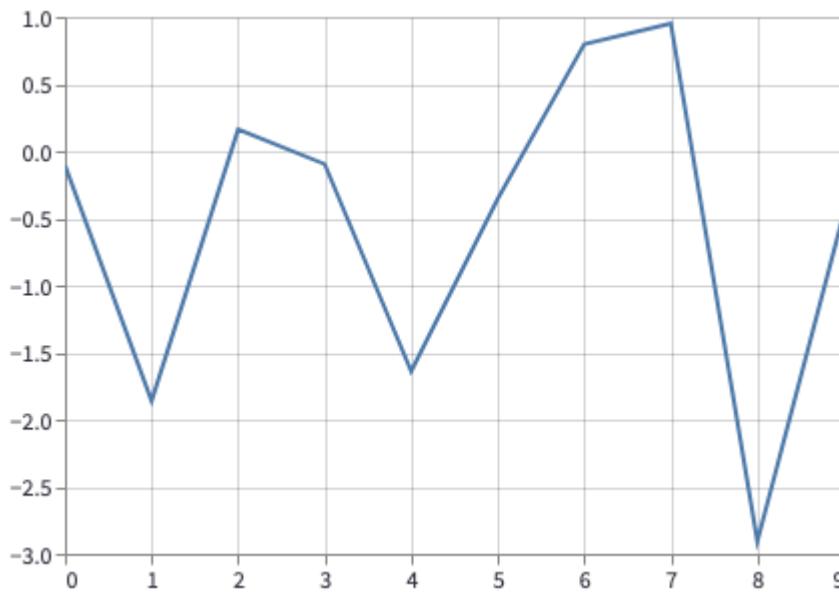
Or you can just call methods directly in the returned objects:

```
col1, col2 = st.columns([3, 1])
data = np.random.randn(10, 1)

col1.subheader("A wide column with a chart")
col1.line_chart(data)

col2.subheader("A narrow column with the data")
col2.write(data)
```

## A wide column with a chart



— 0 ⋮

## A narrow column with the data

	0
0	-0.1063
1	-1.8608
2	0.1628
3	-0.0955
4	-1.6388
5	-0.3542
6	0.7965
7	0.9526
8	-2.9071
9	

([view standalone Streamlit app](#)).

Was this page helpful?

Yes  No

[edit](#) [Suggest edits](#)

[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: st.sidebar](#) [Next: st.expander](#) →

---

[Home](#) [Contact Us](#) [Community](#)



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*add*
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*remove*
    - [st.sidebar](#)
    - [st.columns](#)
    - [st.expander](#)
    - [st.container](#)
    - [st.empty](#)
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [\*cloud\*](#)

### [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notesopen in new](#)
- [Troubleshooting](#)

- [\*school\*](#)

## Knowledge base

- [Tutorials](#)  
*add*
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home/](#)
- [Streamlit library/](#)
- [API reference/](#)
- [Layouts and containers/](#)
- [st.container](#)

## st.container

8

Streamlit Version [v1.9.0](#) ▾

Insert a multi-element container.

Inserts an invisible container into your app that can be used to hold multiple elements. This allows you to, for example, insert multiple elements into your app out of order.

To add elements to the returned container, you can use "with" notation (preferred) or just call methods directly on the returned object. See examples below.

### Function signature

**st.container()**

### Examples

Inserting elements using "with" notation:

```
with st.container():
    st.write("This is inside the container")

    # You can call any Streamlit command, including custom components:
    st.bar_chart(np.random.randn(50, 3))

st.write("This is outside the container")
```

Please wait...

Made with Streamlit

[\(view standalone Streamlit app\)](#)

Inserting elements out of order:

```
container = st.container()  
container.write("This is inside the container")  
st.write("This is outside the container")  
  
# Now insert some more in the container  
container.write("This is inside too")
```

Please wait...

Made with Streamlit

([view standalone Streamlit app](#)).

Was this page helpful?

Yes No  
[edit](#)[Suggest edits](#)  
[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: st.expander](#) [Next: st.empty](#) →

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.

## Documentation

*search*

Search

- *description*

### Streamlit library

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*remove*
    - [st.dataframe](#)
    - [st.table](#)
    - [st.metric](#)
    - [st.json](#)
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*add*
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*add*
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- *cloud*

### Streamlit Cloud

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notes](#)*open in new*
- [Troubleshooting](#)

- *school*

### Knowledge base

- [Tutorials](#)
- [add](#)
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home/](#)
- [Streamlit library/](#)
- [API reference/](#)
- [Data display elements/](#)
- [st.dataframe](#)

## st.dataframe

8

Streamlit Version v1.9.0 ▾

Display a dataframe as an interactive table.

### Function signature

**st.dataframe(data=None, width=None, height=None)**

#### Parameters

data	The data to display. (pandas.DataFrame, pandas.Styler, pyarrow.Table, numpy.ndarray, Iterable, dict, or None)
width (int or None)	Desired width of the UI element expressed in pixels. If None, a default width based on the page width is used.
height (int or None)	Desired height of the UI element expressed in pixels. If None, a default height is used.

#### Examples

```
df = pd.DataFrame(
    np.random.randn(50, 20),
    columns=['col %d' % i for i in range(20))]

st.dataframe(df)  # Same as st.write(df)
```



Please wait...

Made with Streamlit

[\(view standalone Streamlit app\)](#)

```
st.dataframe(df, 200, 100)
```

You can also pass a Pandas Styler object to change the style of the rendered DataFrame:

```
df = pd.DataFrame(  
    np.random.randn(10, 20),  
    columns=['col %d' % i for i in range(20)])  
  
st.dataframe(df.style.highlight_max(axis=0))
```



Please wait...

Made with Streamlit

[\(view standalone Streamlit app\)](#)

Was this page helpful?

Yes  No

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

←[Previous: Data display elements](#)[Next: st.table](#)→

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*remove*
    - [st.button](#)
    - [st.download\\_button](#)
    - [st.checkbox](#)
    - [st.radio](#)
    - [st.selectbox](#)
    - [st.multiselect](#)
    - [st.slider](#)
    - [st.select\\_slider](#)
    - [st.text\\_input](#)
    - [st.number\\_input](#)
    - [st.text\\_area](#)
    - [st.date\\_input](#)
    - [st.time\\_input](#)
    - [st.file\\_uploader](#)
    - [st.camera\\_input](#)
    - [st.color\\_picker](#)
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*add*
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)

- [Cheat sheet](#)

- [cloud](#)

## [Streamlit Cloud](#)

- [Get started](#)
- [add](#)
- [Trust and Security](#)
- [Release notes](#)[open in new](#)
- [Troubleshooting](#)

- [school](#)

## [Knowledge base](#)

- [Tutorials](#)
- [add](#)
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

[Home](#) / [Streamlit library](#) / [API reference](#) / [Input widgets](#) / [\*\*st.date\\_input\*\*](#)

# **st.date\_input**

Streamlit Version

**v1.9.0**

Display a date input widget.

## Function signature

```
st.date_input(label, value=None, min_value=None, max_value=None, key=None, help=None, on_change=None, args=None, kwargs=None, *, disabled=False)
```

## Parameters

label (str)

A short label explaining to the user what this date input is for.

value (datetime.date or datetime.datetime or list/tuple of datetime.date or datetime.datetime or None)

The value of this widget when it first renders. If a list/tuple with 0 to 2 date/datetime values is provided, the datepicker will allow users to provide a range. Defaults to today as a single-date picker.

`min_value` (datetime.date or datetime.datetime)

The minimum selectable date. If value is a date, defaults to value - 10 years. If value is the interval [start, end], defaults to start - 10 years.

`max_value` (datetime.date or datetime.datetime)

The maximum selectable date. If value is a date, defaults to value + 10 years. If value is the interval [start, end], defaults to end + 10 years.

`key` (str or int)

An optional string or integer to use as the unique key for the widget. If this is omitted, a key will be generated for the widget based on its content. Multiple widgets of the same type may not share the same key.

`help` (str)

An optional tooltip that gets displayed next to the input.

`on_change` (callable)

An optional callback invoked when this date\_input's value changes.

`args` (tuple)

An optional tuple of args to pass to the callback.

kwargs (dict)

An optional dict of kwargs to pass to the callback.

disabled (bool)

An optional boolean, which disables the date input if set to True. The default is False. This argument can only be supplied by keyword.

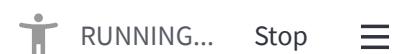
## Returns

(datetime.date or a tuple with 0-2 dates)

The current value of the date input widget.

## Example

```
d = st.date_input(  
    "When's your birthday",  
    datetime.date(2019, 7, 6))  
st.write('Your birthday is:', d)
```



Hosted with Streamlit

[\(view standalone Streamlit app\)](#)

*thumb\_up* Yes

*thumb\_down* No

*edit* [Suggest edits](#)

## Still have questions?

*forum*

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: st.text\\_area](#)

[Next: st.time\\_input](#) →

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*remove*
    - [st.button](#)
    - [st.download\\_button](#)
    - [st.checkbox](#)
    - [st.radio](#)
    - [st.selectbox](#)
    - [st.multiselect](#)
    - [st.slider](#)
    - [st.select\\_slider](#)
    - [st.text\\_input](#)
    - [st.number\\_input](#)
    - [st.text\\_area](#)
    - [st.date\\_input](#)
    - [st.time\\_input](#)
    - [st.file\\_uploader](#)
    - [st.camera\\_input](#)
    - [st.color\\_picker](#)
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*add*
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)

- [Cheat sheet](#)

- [cloud](#)

## [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notes](#)*open in new*
- [Troubleshooting](#)

- [school](#)

## [Knowledge base](#)

- [Tutorials](#)  
*add*
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

[Home](#) / [Streamlit library](#) / [API reference](#) / [Input widgets](#) / [\*\*st.download\\_button\*\*](#)

# **st.download\_button**

8

Streamlit Version

**v1.9.0**

Display a download button widget.

This is useful when you would like to provide a way for your users to download a file directly from your app.

Note that the data to be downloaded is stored in-memory while the user is connected, so it's a good idea to keep file sizes under a couple hundred megabytes to conserve memory.

### Function signature

```
st.download_button(label, data, file_name=None, mime=None, key=None, help=None, on_click=None, args=None, kwargs=None, *, disabled=False)
```

#### Parameters

**label** (str) A short label explaining to the user what this button is for.

**data** (str or bytes or file) The contents of the file to be downloaded. See example below for caching techniques to avoid recomputing this data unnecessarily.

**file\_name** (str) An optional string to use as the name of the file to be downloaded, such as 'my\_file.csv'. If not specified, the name will be automatically generated.

**mime** (str) The MIME type of the data. If None, defaults to "text/plain" (if data is of type *str* or is a textual *file*) or

or None)	"application/octet-stream" (if data is of type <i>bytes</i> or is a binary <i>file</i> ).
key (str or int)	An optional string or integer to use as the unique key for the widget. If this is omitted, a key will be generated for the widget based on its content. Multiple widgets of the same type may not share the same key.
help (str)	An optional tooltip that gets displayed when the button is hovered over.
on_click (callable)	An optional callback invoked when this button is clicked.
args (tuple)	An optional tuple of args to pass to the callback.
kwargs (dict)	An optional dict of kwargs to pass to the callback.
disabled (bool)	An optional boolean, which disables the download button if set to True. The default is False. This argument can only be supplied by keyword.

## Returns

(bool) True if the button was clicked on the last run of the app, False otherwise.

## Examples

Download a large DataFrame as a CSV:

```
@st.cache
def convert_df(df):
    # IMPORTANT: Cache the conversion to prevent computation on every rerun
    return df.to_csv().encode('utf-8')

csv = convert_df(my_large_df)

st.download_button(
    label="Download data as CSV",
    data=csv,
    file_name='large_df.csv',
    mime='text/csv',
)
```

Download a string as a file:

```
text_contents = '''This is some text'''
st.download_button('Download some text', text_contents)
```

Download a binary file:

```
binary_contents = b'example content'
# Defaults to 'application/octet-stream'
st.download_button('Download binary file', binary_contents)
```

Download an image:

```
with open("flower.png", "rb") as file:
    btn = st.download_button(
        label="Download image",
        data=file,
        file_name="flower.png",
```

```
        mime="image/png"
    )
```



Loading...

Loading...

Loading...

[\(view standalone Streamlit app\)](#)

Was this page helpful?

Yes No

[edit](#)[Suggest edits](#)

[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

←[Previous: st.button](#)[Next: st.checkbox](#)→

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.



# st.echo

v1.9.0

Use in a `with` block to draw some code on the app, then execute it.

## Function signature

```
st.echo(code_location="above")
```

## Parameters

**code\_location** ("above" or "below")

Whether to show the echoed code before or after the results of the executed code block.

## Example

```
with st.echo():
    st.write('This code will be printed')
```

## Display code

Sometimes you want your Streamlit app to contain *both* your usual Streamlit graphic elements *and* the code that generated those elements. That's where **st.echo()** comes in.

Ok so let's say you have the following file, and you want to make its app a little bit more self-explanatory by making that middle section visible in the Streamlit app:

```
import streamlit as st

def get_user_name():
    return 'John'
```

```
# -----
# Want people to see this part of the code...

def get_punctuation():
    return '!!!!'

greeting = "Hi there, "
user_name = get_user_name()
punctuation = get_punctuation()

st.write(greeting, user_name, punctuation)

# ...up to here
# -----



foo = 'bar'
st.write('Done!')
```

The file above creates a Streamlit app containing the words "Hi there, **John**", and then "Done!".

Now let's use `st.echo()` to make that middle section of the code visible in the app:

```
import streamlit as st

def get_user_name():
    return 'John'

with st.echo():
    # Everything inside this block will be both printed to the screen
    # and executed.

    def get_punctuation():
        return '!!!!'

    greeting = "Hi there, "
    value = get_user_name()
    punctuation = get_punctuation()

    st.write(greeting, value, punctuation)

# And now we're back to _not_ printing to the screen
```

```
foo = 'bar'  
st.write('Done!')
```

It's *that* simple!

### ★ Note

You can have multiple `st.echo()` blocks in the same file. Use it as often as you wish!

## Was this page helpful?

 Yes     No

 [Suggest edits](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: st.set\\_page\\_config](#)

[Next: st.help](#) →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*add*
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*remove*
    - [st.sidebar](#)
    - [st.columns](#)
    - [st.expander](#)
    - [st.container](#)
    - [st.empty](#)
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [\*cloud\*](#)

### [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notesopen in new](#)
- [Troubleshooting](#)

- [\*school\*](#)

## Knowledge base

- [Tutorials](#)  
*add*
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home/](#)
- [Streamlit library/](#)
- [API reference/](#)
- [Layouts and containers/](#)
- [st.empty](#)

## st.empty

8

Streamlit Version [v1.9.0](#) 

Insert a single-element container.

Inserts a container into your app that can be used to hold a single element. This allows you to, for example, remove elements at any point, or replace several elements at once (using a child multi-element container).

To insert/replace/clear an element on the returned container, you can use "with" notation or just call methods directly on the returned object. See examples below.

### Function signature

`st.empty()`

### Examples

Overwriting elements in-place using "with" notation:

```
import time

with st.empty():
    for seconds in range(60):
        st.write(f"<img alt='hourglass icon' data-bbox='255 665 275 685' style='vertical-align: middle;"/> {seconds} seconds have passed")
        time.sleep(1)
    st.write("✅ 1 minute over!")
```

Replacing several elements, then clearing them:

```
placeholder = st.empty()

# Replace the placeholder with some text:
placeholder.text("Hello")

# Replace the text with a chart:
placeholder.line_chart({"data": [1, 5, 2, 6]})

# Replace the chart with several elements:
with placeholder.container():
    st.write("This is one element")
    st.write("This is another")

# Clear all those elements:
placeholder.empty()
```

Was this page helpful?

[thumb\\_upYes](#) [thumb\\_downNo](#)

[edit](#)[Suggest edits](#)

[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

←[Previous: st.container](#)[Next: Status elements](#)→

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.



# st.error

v1.9.0

Display error message.

## Function signature

```
st.error(body)
```

## Parameters

**body (str)**

The error text to display.

## Example

```
st.error('This is an error')
```



## Was this page helpful?

Yes    No

[Suggest edits](#)



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[\*\*← Previous:\*\* st.snow](#)

**Next:** st.warning →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# st.exception

v1.9.0

Display an exception.

## Function signature

```
st.exception(exception)
```

## Parameters

**exception** (*Exception*)

The exception to display.

## Example

```
e = RuntimeError('This is an exception of type RuntimeError')
st.exception(e)
```



## Was this page helpful?

Yes    No

[Suggest edits](#)



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[\*\*← Previous:\*\* st.success](#)

**Next:** Control flow →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*add*
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*remove*
    - [st.sidebar](#)
    - [st.columns](#)
    - [st.expander](#)
    - [st.container](#)
    - [st.empty](#)
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [\*cloud\*](#)

### [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notesopen in new](#)
- [Troubleshooting](#)

- [\*school\*](#)

## Knowledge base

- [Tutorials](#)  
*add*
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home/](#)
- [Streamlit library/](#)
- [API reference/](#)
- [Layouts and containers/](#)
- [st.expander](#)

## st.expander

8

Streamlit Version [v1.9.0](#) ▾

Insert a multi-element container that can be expanded/collapsed.

Inserts a container into your app that can be used to hold multiple elements and can be expanded or collapsed by the user. When collapsed, all that is visible is the provided label.

To add elements to the returned container, you can use "with" notation (preferred) or just call methods directly on the returned object. See examples below.

### Warning

Currently, you may not put expanders inside another expander.

#### Function signature

**st.expander(label, expanded=False)**

#### Parameters

label (str) A string to use as the header for the expander.

expanded (bool) If True, initializes the expander in "expanded" state. Defaults to False (collapsed).

## Examples

You can use *with* notation to insert any element into an expander

```
st.bar_chart({"data": [1, 5, 2, 6, 2, 1]})

with st.expander("See explanation"):
    st.write("""
        The chart above shows some numbers I picked for you.
        I rolled actual dice for these, so they're *guaranteed* to
        be random.
    """)
    st.image("https://static.streamlit.io/examples/dice.jpg")
```

Loading...

See explanation

+

Made with Streamlit

[\(view standalone Streamlit app\)](#)

Or you can just call methods directly in the returned objects:

```
st.bar_chart({"data": [1, 5, 2, 6, 2, 1]})

expander = st.expander("See explanation")
expander.write("""
    The chart above shows some numbers I picked for you.
    I rolled actual dice for these, so they're *guaranteed* to
    be random.
""")
expander.image("https://static.streamlit.io/examples/dice.jpg")
```

Loading...

See explanation

+

Made with Streamlit

[\(view standalone Streamlit app\)](#)

Or you can use object notation and just call methods directly in the returned objects:

```
import streamlit as st

st.bar_chart({"data": [1, 5, 2, 6, 2, 1]})

expander = st.expander("See explanation")
expander.write("""
    The chart above shows some numbers I picked for you.
    I rolled actual dice for these, so they're *guaranteed* to
    be random.
""")
expander.image("https://static.streamlit.io/examples/dice.jpg")
```

Was this page helpful?

Yes  No

[edit](#)[Suggest edits](#)

[forum](#)

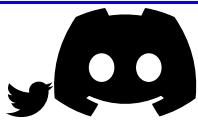
## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

[← Previous: st.columns](#) [Next: st.container →](#)

---

[Home](#) [Contact Us](#) [Community](#)



Copyright © 2022, Streamlit Inc.



## ! Important

This is an experimental feature. Experimental features and their APIs may change or be removed at any time. To learn more, click [here](#).

# st.experimental\_memo

v1.9.0

Function decorator to memoize function executions.

Memoized data is stored in "pickled" form, which means that the return value of a memoized function must be pickleable.

Each caller of a memoized function gets its own copy of the cached data.

You can clear a memoized function's cache with `f.clear()`.

## Function signature

```
st.experimental_memo(func=None, *, persist=None, show_spinner=True,  
suppress_st_warning=False, max_entries=None, ttl=None)
```

## Parameters

### `func (callable)`

The function to memoize. Streamlit hashes the function's source code.

### `persist (str or None)`

Optional location to persist cached data to. Currently, the only valid value is "disk", which will persist to the local disk.

### **show\_spinner (boolean)**

Enable the spinner. Default is True to show a spinner when there is a cache miss.

### **suppress\_st\_warning (boolean)**

Suppress warnings about calling Streamlit functions from within the cached function.

### **max\_entries (int or None)**

The maximum number of entries to keep in the cache, or None for an unbounded cache. (When a new entry is added to a full cache, the oldest cached entry will be removed.) The default is None.

### **ttl (float or None)**

The maximum number of seconds to keep an entry in the cache, or None if cache entries should not expire. The default is None.

## **Example**

```
@st.experimental_memo
def fetch_and_clean_data(url):
    # Fetch data from URL here, and then clean it up.
    return data

d1 = fetch_and_clean_data(DATA_URL_1)
# Actually executes the function, since this is the first time it was
# encountered.

d2 = fetch_and_clean_data(DATA_URL_1)
# Does not execute the function. Instead, returns its previously computed
# value. This means that now the data in d1 is the same as in d2.

d3 = fetch_and_clean_data(DATA_URL_2)
# This is a different URL, so the function executes.
```

To set the `persist` parameter, use this command as follows:

```
@st.experimental_memo(persist="disk")
def fetch_and_clean_data(url):
    # Fetch data from URL here, and then clean it up.
    return data
```

By default, all parameters to a memoized function must be hashable. Any parameter whose name begins with `_` will not be hashed. You can use this as an "escape hatch" for parameters that are not hashable:

```
@st.experimental_memo
def fetch_and_clean_data(_db_connection, num_rows):
    # Fetch data from _db_connection here, and then clean it up.
    return data

connection = make_database_connection()
d1 = fetch_and_clean_data(connection, num_rows=10)
# Actually executes the function, since this is the first time it was
# encountered.

another_connection = make_database_connection()
d2 = fetch_and_clean_data(another_connection, num_rows=10)
# Does not execute the function. Instead, returns its previously computed
# value – even though the _database_connection parameter was different
# in both calls.
```

A memoized function's cache can be procedurally cleared:

```
@st.experimental_memo
def fetch_and_clean_data(_db_connection, num_rows):
    # Fetch data from _db_connection here, and then clean it up.
    return data

fetch_and_clean_data.clear()
# Clear all cached entries for this function.
```

 Yes

 No

 Suggest edits

---

## Still have questions?

 Our [forums](#) are full of helpful information and Streamlit experts.

---

← **Previous:** st.cache

**Next:** Clear memo →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



## ! Important

This is an experimental feature. Experimental features and their APIs may change or be removed at any time. To learn more, click [here](#).

# st.experimental\_singleton

v1.9.0

Function decorator to store singleton objects.

Each singleton object is shared across all users connected to the app. Singleton objects *must* be thread-safe, because they can be accessed from multiple threads concurrently.

(If thread-safety is an issue, consider using `st.session_state` to store per-session singleton objects instead.)

You can clear a memoized function's cache with `f.clear()`.

## Function signature

```
st.experimental_singleton(func=None, *, show_spinner=True,  
suppress_st_warning=False)
```

## Parameters

### `func (callable)`

The function that creates the singleton. Streamlit hashes the function's source code.

### `show_spinner (boolean)`

Enable the spinner. Default is True to show a spinner when there is a "cache miss" and the singleton is being created.

## suppress\_st\_warning (boolean)

Suppress warnings about calling Streamlit functions from within the singleton function.

## Example

```
@st.experimental_singleton
def get_database_session(url):
    # Create a database session object that points to the URL.
    return session

s1 = get_database_session(SESSION_URL_1)
# Actually executes the function, since this is the first time it was
# encountered.

s2 = get_database_session(SESSION_URL_1)
# Does not execute the function. Instead, returns its previously computed
# value. This means that now the connection object in s1 is the same as in s2.

s3 = get_database_session(SESSION_URL_2)
# This is a different URL, so the function executes.
```

By default, all parameters to a singleton function must be hashable. Any parameter whose name begins with `_` will not be hashed. You can use this as an "escape hatch" for parameters that are not hashable:

```
@st.experimental_singleton
def get_database_session(_sessionmaker, url):
    # Create a database connection object that points to the URL.
    return connection

s1 = get_database_session(create_sessionmaker(), DATA_URL_1)
# Actually executes the function, since this is the first time it was
# encountered.

s2 = get_database_session(create_sessionmaker(), DATA_URL_1)
# Does not execute the function. Instead, returns its previously computed
# value – even though the _sessionmaker parameter was different
# in both calls.
```

A singleton function's cache can be procedurally cleared:



```
@st.experimental_singleton  
def get_database_session(_sessionmaker, url):  
    # Create a database connection object that points to the URL.  
    return connection  
  
get_database_session.clear()  
# Clear all cached entries for this function.
```

## Was this page helpful?

Yes

No

[Suggest edits](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

[← Previous: Clear memo](#)

[Next: Clear singleton →](#)

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*remove*
    - [st.button](#)
    - [st.download\\_button](#)
    - [st.checkbox](#)
    - [st.radio](#)
    - [st.selectbox](#)
    - [st.multiselect](#)
    - [st.slider](#)
    - [st.select\\_slider](#)
    - [st.text\\_input](#)
    - [st.number\\_input](#)
    - [st.text\\_area](#)
    - [st.date\\_input](#)
    - [st.time\\_input](#)
    - [st.file\\_uploader](#)
    - [st.camera\\_input](#)
    - [st.color\\_picker](#)
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*add*
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)

- [Cheat sheet](#)

- [cloud](#)

## [Streamlit Cloud](#)

- [Get started](#)
- *add*
- [Trust and Security](#)
- [Release notes](#) *open in new*
- [Troubleshooting](#)

- [school](#)

## [Knowledge base](#)

- [Tutorials](#)
- *add*
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home](#)/
- [Streamlit library](#)/
- [API reference](#)/
- [Input widgets](#)/
- [st.file\\_uploader](#)

## st.file\_uploader

8

Streamlit Version [v1.9.0](#) ▾

Display a file uploader widget.

By default, uploaded files are limited to 200MB. You can configure this using the `server.maxUploadSize` config option. For more info on how to set config options, see <https://docs.streamlit.io/library/advanced-features/configuration#set-configuration-options>

### Function signature

```
st.file_uploader(label, type=None, accept_multiple_files=False, key=None, help=None, on_change=None,
                 args=None, kwargs=None, *, disabled=False)
```

#### Parameters

label (str)	A short label explaining to the user what this file uploader is for.
-------------	--

type (str or list of str)	Array of allowed extensions. ['png', 'jpg'] The default is None, which means all extensions are allowed.
---------------------------	--

accept_multiple_files (bool)	If True, allows the user to upload multiple files at the same time, in which case the return value will be a list of files. Default: False
------------------------------	--

key (str or int)	An optional string or integer to use as the unique key for the widget. If this is omitted, a key will be generated for the widget based on its content. Multiple widgets of the same type may not share the same key.
------------------	---

help (str)	A tooltip that gets displayed next to the file uploader.
on_change (callable)	An optional callback invoked when this file_uploader's value changes.
args (tuple)	An optional tuple of args to pass to the callback.
kwargs (dict)	An optional dict of kwargs to pass to the callback.
disabled (bool)	An optional boolean, which disables the file uploader if set to True. The default is False. This argument can only be supplied by keyword.
Returns	
(None or UploadedFile or list of UploadedFile)	<ul style="list-style-type: none"> <li>• If accept_multiple_files is False, returns either None or an UploadedFile object.</li> <li>• If accept_multiple_files is True, returns a list with the uploaded files as UploadedFile objects. If no files were uploaded, returns an empty list.</li> </ul> <p>The UploadedFile class is a subclass of BytesIO, and therefore it is "file-like". This means you can pass them anywhere where a file is expected.</p>

## Examples

Insert a file uploader that accepts a single file at a time:

```
uploaded_file = st.file_uploader("Choose a file")
if uploaded_file is not None:
    # To read file as bytes:
    bytes_data = uploaded_file.getvalue()
    st.write(bytes_data)

    # To convert to a string based IO:
    stringio = StringIO(uploaded_file.getvalue().decode("utf-8"))
    st.write(stringio)

    # To read file as string:
    string_data = stringio.read()
    st.write(string_data)

    # Can be used wherever a "file-like" object is accepted:
    dataframe = pd.read_csv(uploaded_file)
    st.write(dataframe)
```

Insert a file uploader that accepts multiple files at a time:

```
uploaded_files = st.file_uploader("Choose a CSV file", accept_multiple_files=True)
for uploaded_file in uploaded_files:
    bytes_data = uploaded_file.read()
    st.write("filename:", uploaded_file.name)
    st.write(bytes_data)
```

Choose a file



Drag and drop files here

Limit 200MB per file

[Browse files](#)

[\(view standalone Streamlit app\)](#).

Was this page helpful?

Yes    No

[edit](#)[Suggest edits](#)

[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

←[Previous: st.time\\_input](#)[Next: st.camera\\_input](#)→

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.



- [Home/](#)
- [Streamlit library/](#)
- [API reference/](#)
- [Control flow/](#)
- [st.form](#)

## st.form

Streamlit Version v1.9.0 ▾

Create a form that batches elements together with a "Submit" button.

A form is a container that visually groups other elements and widgets together, and contains a Submit button. When the form's Submit button is pressed, all widget values inside the form will be sent to Streamlit in a batch.

To add elements to a form object, you can use "with" notation (preferred) or just call methods directly on the form. See examples below.

Forms have a few constraints:

- Every form must contain a `st.form_submit_button`.
- `st.button` and `st.download_button` cannot be added to a form.
- Forms can appear anywhere in your app (sidebar, columns, etc), but they cannot be embedded inside other forms.

For more information about forms, check out our [blog post](#).

### Function signature

```
st.form(key, clear_on_submit=False)
```

### Parameters

key (str)

A string that identifies the form. Each form must have its own key. (This key is not displayed to the user in the interface.)

clear\_on\_submit (bool)

If True, all widgets inside the form will be reset to their default values after the user presses the Submit button. Defaults to False. (Note that Custom Components are unaffected by this flag, and will not be reset to their defaults on form submission.)

## Examples

Inserting elements using "with" notation:

```
with st.form("my_form"):
    st.write("Inside the form")
    slider_val = st.slider("Form slider")
    checkbox_val = st.checkbox("Form checkbox")

    # Every form must have a submit button.
    submitted = st.form_submit_button("Submit")
    if submitted:
        st.write("slider", slider_val, "checkbox", checkbox_val)

st.write("Outside the form")
```

[Copy](#)

Inserting elements out of order:

```
form = st.form("my_form")
form.slider("Inside the form")
st.slider("Outside the form")

# Now add a submit button to the form:
form.form_submit_button("Submit")
```

[Copy](#)

## Was this page helpful?

*thumb\_up* Yes      *thumb\_down* No

*edit* [Suggest edits](#)

Still have questions?

*forum*

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: st.stop](#)

[Next: st.form\\_submit\\_button](#) →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*add*
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*add*
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*remove*
    - [st.stop](#)
    - [st.form](#)
    - [st.form\\_submit\\_button](#)
    - [st.experimental\\_rerun](#)
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [\*cloud\*](#)

### [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notesopen in new](#)
- [Troubleshooting](#)

- [\*school\*](#)

### [Knowledge base](#)

- [Tutorials](#)
- [add](#)
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home/](#)
- [Streamlit library/](#)
- [API reference/](#)
- [Control flow/](#)
- [st.form\\_submit\\_button](#)

## st.form\_submit\_button

8

Streamlit Version v1.9.0 ▾

Display a form submit button.

When this button is clicked, all widget values inside the form will be sent to Streamlit in a batch.

Every form must have a `form_submit_button`. A `form_submit_button` cannot exist outside a form.

For more information about forms, check out our [blog post](#).

### Function signature

`st.form_submit_button(label="Submit", help=None, on_click=None, args=None, kwargs=None)`

#### Parameters

`label` (str) A short label explaining to the user what this button is for. Defaults to "Submit".

`help` (str or None) A tooltip that gets displayed when the button is hovered over. Defaults to None.

`on_click` (callable) An optional callback invoked when this button is clicked.

`args` (tuple) An optional tuple of args to pass to the callback.

`kwargs` (dict) An optional dict of kwargs to pass to the callback.

#### Returns

(bool) True if the button was clicked.

Was this page helpful?

Yes  No

[edit](#) [Suggest edits](#)

[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.



Copyright © 2022, Streamlit Inc.



# `st.graphviz_chart`

Streamlit Version

**v1.9.0**

Display a graph using the dagre-d3 library.

## Function signature

```
st.graphviz_chart(figure_or_dot, use_container_width=False)
```

## Parameters

`figure_or_dot` (graphviz.dot.Graph, graphviz.dot.Digraph, str)

The Graphlib graph object or dot string to display

`use_container_width` (bool)

If True, set the chart width to the column width. This takes precedence over the figure's native `width` value.

## Example

```
import streamlit as st
import graphviz as graphviz

# Create a graphlib graph object
graph = graphviz.Digraph()
graph.edge('run', 'intr')
graph.edge('intr', 'runbl')
graph.edge('runbl', 'run')
graph.edge('run', 'kernel')
graph.edge('kernel', 'zombie')
graph.edge('kernel', 'sleep')
graph.edge('kernel', 'runmem')
graph.edge('sleep', 'swap')
graph.edge('swap', 'runswap')
graph.edge('runswap', 'new')
```

```
graph.edge('runswap', 'runmem')
graph.edge('new', 'runmem')
graph.edge('sleep', 'runmem')

st.graphviz_chart(graph)
```

Or you can render the chart from the graph using GraphViz's Dot language:

```
st.graphviz_chart('''
    digraph {
        run -> intr
        intr -> runbl
        runbl -> run
        run -> kernel
        kernel -> zombie
        kernel -> sleep
        kernel -> runmem
        sleep -> swap
        swap -> runswap
        runswap -> new
        runswap -> runmem
        new -> runmem
        sleep -> runmem
    }
''')
```



Please wait...

Made with Streamlit

[\(view standalone Streamlit app\)](#)

*thumb\_up* Yes

*thumb\_down* No

*edit* [Suggest edits](#)

## Still have questions?

*forum*

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: st.pydeck\\_chart](#)

[Next: st.map](#) →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# st.header

v1.9.0

Display text in header formatting.

## Function signature

```
st.header(body, anchor=None)
```

## Parameters

### body (*str*)

The text to display.

### anchor (*str*)

The anchor name of the header that can be accessed with #anchor in the URL. If omitted, it generates an anchor using the body.

## Example

```
st.header('This is a header')
```



# This is a header

 [Suggest edits](#)

---

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[!\[\]\(094fefb3f63c76c8fc5310b7fb0640f6\_img.jpg\) Previous:](#) st.title

[Next:](#) st.subheader [!\[\]\(69c0fc4b2bc0f479656199b11e24948f\_img.jpg\)](#)

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# st.help

**v1.9.0**

Display object's doc string, nicely formatted.

Displays the doc string for this object.

## Function signature

```
st.help(obj)
```

## Parameters

**obj (Object)**

The object whose docstring should be displayed.

## Example

Don't remember how to initialize a dataframe? Try this:

```
st.help(pandas.DataFrame)
```

Want to quickly check what datatype is output by a certain function? Try:

```
x = my_poorly_documented_function()  
st.help(x)
```

Was this page helpful?

Yes

No

 Suggest edits

---

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[← Previous: st.echo](#)

[Next: st.experimental\\_show →](#)

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.

## Documentation

*search*

Search

- *description*

### Streamlit library

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*add*
  - [Media elements](#)  
*remove*
    - [st.image](#)
    - [st.audio](#)
    - [st.video](#)
  - [Layouts and containers](#)  
*add*
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- *cloud*

### Streamlit Cloud

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notes](#)*open in new*
- [Troubleshooting](#)

- *school*

### Knowledge base

- [Tutorials](#)
- [add](#)
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home](#)/
- [Streamlit library](#)/
- [API reference](#)/
- [Media elements](#)/
- [st.image](#)

## st.image

8

Streamlit Version v1.9.0 ▾

Display an image or list of images.

### Function signature

```
st.image(image, caption=None, width=None, use_column_width=None, clamp=False, channels="RGB",
         output_format="auto")
```

#### Parameters

image (numpy.ndarray, [numpy.ndarray], BytesIO, str, or [str])	Monochrome image of shape (w,h) or (w,h,1) OR a color image of shape (w,h,3) OR an RGBA image of shape (w,h,4) OR a URL to fetch the image from OR a path of a local image file OR an SVG XML string like <svg xmlns=...></svg> OR a list of one of the above, to display multiple images.
caption (str or list of str)	Image caption. If displaying multiple images, caption should be a list of captions (one for each image).
width (int or None)	Image width. None means use the image width, but do not exceed the width of the column. Should be set for SVG images, as they have no default image width.
use_column_width ('auto' or 'always' 'never' or bool)	If 'auto', set the image's width to its natural size, but do not exceed the width of the column. If ('auto' or 'always' or True, set the image's width to the column width. If 'never' or False, set the image's width to its natural size. Note: if set, <i>use_column_width</i> takes precedence over the <i>width</i> parameter.
clamp (bool)	Clamp image pixel values to a valid range ([0-255] per channel). This is only meaningful for byte array images; the parameter is ignored for image URLs. If this is not set, and an image has an out-of-range value, an error will be thrown.
channels ('RGB' or 'BGR')	If image is an nd.array, this parameter denotes the format used to represent color information. Defaults to 'RGB', meaning <i>image[:, :, 0]</i> is the red channel, <i>image[:, :, 1]</i> is green, and <i>image[:, :, 2]</i> is blue. For images coming from libraries like OpenCV you should set this to 'BGR', instead.
output_format ('JPEG', 'PNG', or 'auto')	This parameter specifies the format to use when transferring the image data. Photos should use the JPEG format for lossy compression while diagrams should use the PNG format for lossless compression. Defaults to 'auto' which identifies the compression type based on the type and format of the image argument.

## Example

```
from PIL import Image
image = Image.open('sunrise.jpg')

st.image(image, caption='Sunrise by the mountains')
```



Sunrise by the mountains

## Image credit:

Creator: User *Neil Iris (@neil\_ingham)* from *Unsplash*

License: Do whatever you want. <https://unsplash.com/license>

URL: <https://unsplash.com/photos/I2UR7wEftf4>

Made with Streamlit

([view standalone Streamlit app](#)).

Was this page helpful?

Yes No

[edit](#)[Suggest edits](#)  
[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: Media elements](#) [Next: st.audio](#) →



Copyright © 2022, Streamlit Inc.



# st.info

v1.9.0

Display an informational message.

## Function signature

```
st.info(body)
```

## Parameters

**body** (*str*)

The info text to display.

## Example

```
st.info('This is a purely informational message')
```



## Was this page helpful?

👍 Yes

👎 No

[Suggest edits](#)



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

← **Previous:** st.warning

**Next:** st.success →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*remove*
    - [st.dataframe](#)
    - [st.table](#)
    - [st.metric](#)
    - [st.json](#)
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*add*
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*add*
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [\*cloud\*](#)

### [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notesopen in new](#)
- [Troubleshooting](#)

- [\*school\*](#)

### [Knowledge base](#)

- [Tutorials](#)
- [add](#)
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home](#)/
- [Streamlit library](#)/
- [API reference](#)/
- [Data display elements](#)/
- [st.json](#)

## st.json

8

Streamlit Version v1.9.0 ▾

Display object or string as a pretty-printed JSON string.

**Function signature**

**st.json(body, \*, expanded=True)**

### Parameters

body (Object or str)	The object to print as JSON. All referenced objects should be serializable to JSON as well. If object is a string, we assume it contains serialized JSON.
expanded (bool)	An optional boolean that allows the user to set whether the initial state of this json element should be expanded. Defaults to True. This argument can only be supplied by keyword.

### Example

```
st.json({
    'foo': 'bar',
    'baz': 'boz',
    'stuff': [
        'stuff 1',
        'stuff 2',
        'stuff 3',
        'stuff 5',
    ],
})
```

```
▼ {  
  "foo" : "bar"  
  "baz" : "boz"  
  ▼ "stuff" : [  
    0 : "stuff 1"  
    1 : "stuff 2"  
    2 : "stuff 3"  
    3 : "stuff 5"  
  ]  
}
```

([view standalone Streamlit app](#))

Was this page helpful?

Yes  No

[edit](#)[Suggest edits](#)

[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: st.metric](#) [Next: Chart elements](#) →

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.



# Documentation



Search



## Streamlit library

- Get started



- API reference



- Write and magic



- Text elements



- st.markdown

- st.title

- st.header

- st.subheader

- st.caption

- st.code

- st.text

- st.latex

- Data display elements



- Chart elements



- Input widgets

+

- Media elements

+

- Layouts and containers

+

- Status elements

+

- Control flow

+

- Utilities

+

- Mutate charts

- State management

- Performance

+

- Advanced features

+

- Components

+

- Changelog

- Cheat sheet



- Streamlit Cloud

- Get started

+

- Trust and Security

- Release notes

- Troubleshooting



- Tutorials
  - +
- Using Streamlit
- Streamlit Components
- Installing dependencies
- Deployment issues

Home / Streamlit library / API reference / Text elements / **st.latex**

# st.latex



## v1.9.0

Display mathematical expressions formatted as LaTeX.

Supported LaTeX functions are listed at <https://katex.org/docs/supported.html>.

### Function signature

`st.latex(body)`

#### Parameters

**body (str or SymPy)** The string or SymPy expression to display as LaTeX. If str, it's a good idea to use raw expression) Python strings since LaTeX uses backslashes a lot.

#### Example

```
st.latex(r"""
    a + ar + a r^2 + a r^3 + \cdots + a r^{n-1} =
    \sum_{k=0}^{n-1} ar^k =
    a \left(\frac{1-r^n}{1-r}\right)
""")
```

$$a + ar + ar^2 + ar^3 + \cdots + ar^{n-1} = \sum_{k=0}^{n-1} ar^k = a \left( \frac{1 - r^n}{1 - r} \right)$$

 Yes  No

 Suggest edits



## Still have questions?

Our forums are full of helpful information and Streamlit experts.



Previous: st.textNext: Data display elements



[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.



# Documentation



Search



Streamlit library

- Get started



- API reference



- Write and magic



- Text elements



- Data display elements



- Chart elements



- st.line\_chart

- st.area\_chart

- st.bar\_chart

- st.pyplot

- st.altair\_chart

- st.vega\_lite\_chart

- st.plotly\_chart

- st.bokeh\_chart

- st.pydeck\_chart

- st.graphviz\_chart

- st.map

- Input widgets

+

- Media elements

+

- Layouts and containers

+

- Status elements

+

- Control flow

+

- Utilities

+

- Mutate charts

- State management

- Performance

+

- Advanced features

+

- Components

+

- Changelog

- Cheat sheet



## Streamlit Cloud

- Get started

+

- Trust and Security



- Release notes

- Troubleshooting



Knowledge base

- Tutorials

+

- Using Streamlit

- Streamlit Components

- Installing dependencies

- Deployment issues

[Home](#) / [Streamlit library](#) / [API reference](#) / [Chart elements](#) / [\*\*st.line\\_chart\*\*](#)

# **st.line\_chart**

v1.9.0

Display a line chart.

This is syntax-sugar around `st.altair_chart`. The main difference is this command uses the data's own column and indices to figure out the chart's spec. As a result this is easier to use for many "just plot this" scenarios, while being less customizable.

If `st.line_chart` does not guess the data specification correctly, try specifying your desired chart using `st.altair_chart`.

## Function signature

```
st.line_chart(data=None, width=0, height=0, use_container_width=True)
```

Parameters

**data** (*pandas.DataFrame*,  
*pandas.Styler*,  
*pyarrow.Table*,  
*numpy.ndarray*, *Iterable*,  
*dict or None*)

**width** (*int*)

**height** (*int*)

**use\_container\_width** (*bool*)

Data to be plotted. Pyarrow tables are not supported by Streamlit's legacy DataFrame serialization (i.e. with `config.dataFrameSerialization = "legacy"`). To use pyarrow tables, please enable pyarrow by changing the config setting, `config.dataFrameSerialization = "arrow"`.

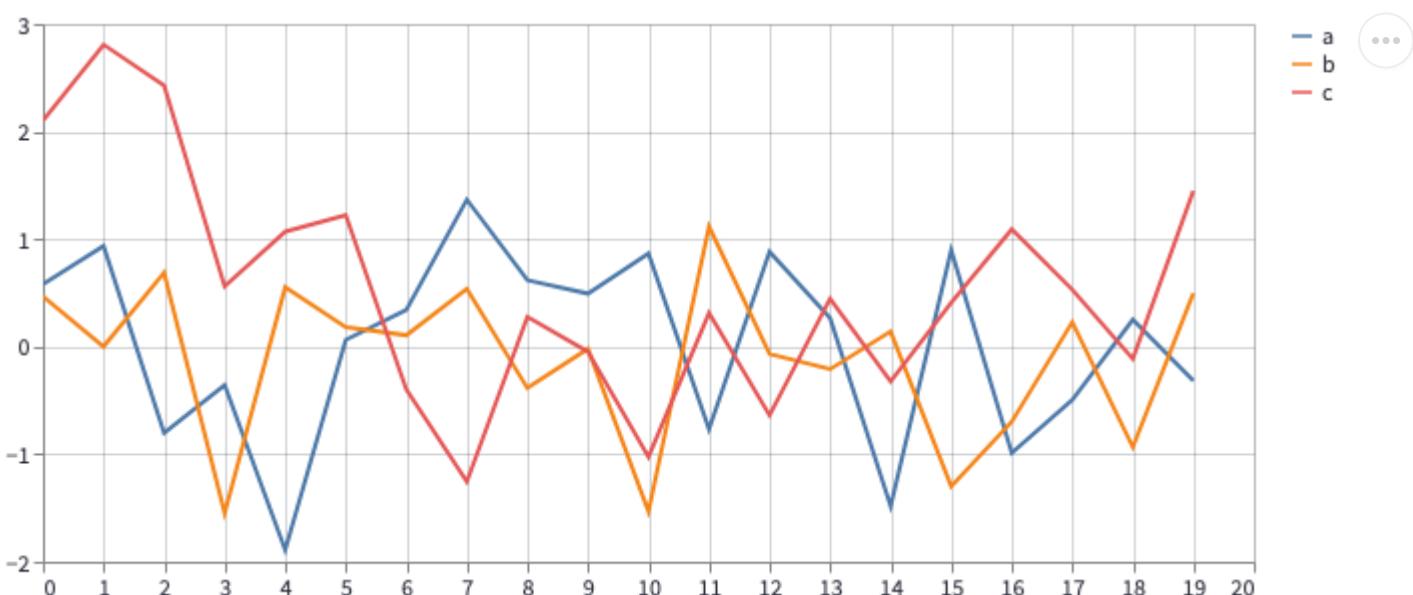
The chart width in pixels. If 0, selects the width automatically.

The chart height in pixels. If 0, selects the height automatically.

If True, set the chart width to the column width. This takes precedence over the width argument.

# Example

```
chart_data = pd.DataFrame(  
    np.random.randn(20, 3),  
    columns=['a', 'b', 'c'])  
  
st.line_chart(chart_data)
```



(view standalone Streamlit app).

Was this page helpful?

Yes No

Suggest edits



## Still have questions?

Our forums are full of helpful information and Streamlit experts.



Previous: Chart elements

Next: st.area\_chart



[Home](#) [Contact Us](#) [Community](#)



Copyright © 2022, Streamlit Inc.



# st.map

v1.9.0

Display a map with points on it.

This is a wrapper around st.pydeck\_chart to quickly create scatterplot charts on top of a map, with auto-centering and auto-zoom.

When using this command, we advise all users to use a personal Mapbox token. This ensures the map tiles used in this chart are more robust. You can do this with the mapbox.token config option.

To get a token for yourself, create an account at <https://mapbox.com>. It's free! (for moderate usage levels). For more info on how to set config options, see <https://docs.streamlit.io/library/advanced-features/configuration#set-configuration-options>

## Function signature

```
st.map(data=None, zoom=None, use_container_width=True)
```

## Parameters

**data** (*pandas.DataFrame*, *pandas.Styler*, *numpy.ndarray*, *Iterable*, *dict*,)

or None The data to be plotted. Must have columns called 'lat', 'lon', 'latitude', or 'longitude'.

**zoom** (*int*)

Zoom level as specified in [https://wiki.openstreetmap.org/wiki/Zoom\\_levels](https://wiki.openstreetmap.org/wiki/Zoom_levels)

## Example

```
import streamlit as st
import pandas as pd
```



```
import numpy as np

df = pd.DataFrame(
    np.random.randn(1000, 2) / [50, 50] + [37.76, -122.4],
    columns=['lat', 'lon'])

st.map(df)
```



([view standalone Streamlit app](#)).

## Was this page helpful?

 Yes     No

 [Suggest edits](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: st.graphviz\\_chart](#)

[Next: Input widgets](#) →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



- Home/
- Streamlit library/
- API reference/
- Text elements/
- st.markdown

## st.markdown

v1.9.0 ▾

Display string formatted as Markdown.

### Function signature

```
st.markdown(body, unsafe_allow_html=False)
```

### Parameters

#### body (str)

The string to display as Github-flavored Markdown. Syntax information can be found at:

<https://github.github.com/gfm>.

This also supports:

- Emoji shortcodes, such as :+1: and :sunglasses:. For a list of all supported codes, see <https://share.streamlit.io/streamlit/emoji-shortcodes>.
- LaTeX expressions, by wrapping them in "\$" or "\$\$" (the "\$\$" must be on their own lines). Supported LaTeX functions are listed at <https://katex.org/docs/supported.html>.

#### unsafe\_allow\_html (bool)

By default, any HTML tags found in the body will be escaped and therefore treated as pure text. This behavior may be turned off by setting this argument to True.

That said, we *strongly advise against it*. It is hard to write secure HTML, so by using this argument you may be compromising your users' security. For more information, see:

<https://github.com/streamlit/streamlit/issues/152>

Also note that `unsafe\_allow\_html` is a temporary measure and may be removed from Streamlit at any time.

If you decide to turn on HTML anyway, we ask you to please tell us your exact use case here:

<https://discuss.streamlit.io/t/96>

This will help us come up with safe APIs that allow you to do what you want.

## Example

```
st.markdown('Streamlit is **_really_ cool**.')
```

Copy

Streamlit is *really* cool.

## Was this page helpful?

 Yes     No

 [Suggest edits](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

 [Previous: Text elements](#)

[Next: st.title](#) 



Copyright © 2022, Streamlit Inc.

## Documentation

*search*

Search

- *description*

### Streamlit library

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*remove*
    - [st.dataframe](#)
    - [st.table](#)
    - [st.metric](#)
    - [st.json](#)
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*add*
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*add*
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- *cloud*

### Streamlit Cloud

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notes](#)*open in new*
- [Troubleshooting](#)

- *school*

### Knowledge base

- [Tutorials](#)
- [add](#)
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home/](#)
- [Streamlit library/](#)
- [API reference/](#)
- [Data display elements/](#)
- [st.metric](#)

## st.metric

8

Streamlit Version [v1.9.0](#) ▾

Display a metric in big bold font, with an optional indicator of how the metric changed.

Tip: If you want to display a large number, it may be a good idea to shorten it using packages like [millify](#) or [numerize](#). E.g. 1234 can be displayed as 1.2k using `st.metric("Short number", millify(1234))`.

### Function signature

```
st.metric(label, value, delta=None, delta_color="normal")
```

#### Parameters

**label** (str) The header or Title for the metric

**value** (int, float, str, or None) Value of the metric. None is rendered as a long dash.

**delta** (int, float, str, or None) Indicator of how the metric changed, rendered with an arrow below the metric. If delta is negative (int/float or starts with a minus sign (str), the arrow points down and the text is red; else the arrow points up and the text is green. If None (default), no delta indicator is shown.

**delta\_color** (str) If "normal" (default), the delta indicator is shown as described above. If "inverse", it is red when positive and green when negative. This is useful when a negative change is considered good, e.g. if cost decreased. If "off", delta is shown in gray regardless of its value.

#### Example

```
st.metric(label="Temperature", value="70 °F", delta="1.2 °F")
```

Temperature

70 °F

↑ 1.2 °F

[\(view standalone Streamlit app\)](#)

`st.metric` looks especially nice in combination with `st.columns`:

```
col1, col2, col3 = st.columns(3)
col1.metric("Temperature", "70 °F", "1.2 °F")
col2.metric("Wind", "9 mph", "-8%")
col3.metric("Humidity", "86%", "4%)
```

Please wait...

Hosted with Streamlit

[\(view standalone Streamlit app\)](#)

The delta indicator color can also be inverted or turned off:

```
st.metric(label="Gas price", value=4, delta=-0.5,
          delta_color="inverse")
st.metric(label="Active developers", value=123, delta=123,
          delta_color="off")
```

Please wait...

[\(view standalone Streamlit app\)](#)

[thumb\\_upYes](#) [thumb\\_downNo](#)

[edit](#)[Suggest edits](#)

[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

←[Previous: st.table](#)[Next: st.json](#)→

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.



# st.multiselect

8

## v1.9.0

Display a multiselect widget.

The multiselect widget starts as empty.

### Function signature

```
st.multiselect(label, options, default=None, format_func=special_internal_function,  
key=None, help=None, on_change=None, args=None, kwargs=None, *, disabled=False)
```

#### Parameters

<b>label</b> (str)	A short label explaining to the user what this select widget is for.
<b>options</b> (Sequence[V], <i>numpy.ndarray</i> , <i>pandas.Series</i> , <i>pandas.DataFrame</i> , or <i>pandas.Index</i> )	Labels for the select options. This will be cast to str internally by default. For pandas.DataFrame, the first column is selected.
<b>default</b> ([V], V, or None)	List of default values. Can also be a single value.
<b>format_func</b> (function)	Function to modify the display of selectbox options. It receives the raw option as an argument and should output the label to be shown for that option. This has no impact on the return value of the multiselect.
<b>key</b> (str or int)	An optional string or integer to use as the unique key for the widget. If this is omitted, a key will be generated for the widget based on its content. Multiple widgets of the same type may not share the same key.
<b>help</b> (str)	An optional tooltip that gets displayed next to the multiselect.
<b>on_change</b> (callable)	An optional callback invoked when this multiselect's value changes.
<b>args</b> (tuple)	An optional tuple of args to pass to the callback.
<b>kwargs</b> (dict)	An optional dict of kwargs to pass to the callback.
<b>disabled</b> (bool)	An optional boolean, which disables the multiselect widget if set to True. The default is False. This argument can only be supplied by keyword.
<b>Returns</b>	
(list)	A list with the selected options

#### Example

```
options = st.multiselect(  
    'What are your favorite colors',  
    ['Green', 'Yellow', 'Red', 'Blue'],  
    ['Yellow', 'Red'])  
  
st.write('You selected:', options)
```



What are your favorite colors

Yellow  Red ✖ ▾

You selected:

```
▼ [  
  |  0 : "Yellow"  
  |  1 : "Red"  
  ]
```

[\(view standalone Streamlit app\)](#)

## Note

User experience can be degraded for large lists of *options* (100+), as this widget is not designed to handle arbitrary text search efficiently. See this [thread](#) on the Streamlit community forum for more information and [GitHub issue #1059](#) for updates on the issue.

Was this page helpful?

Yes No

Suggest edits



## Still have questions?

Our forums are full of helpful information and Streamlit experts.





---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# st.number\_input

v1.9.0

Display a numeric input widget.

## Function signature

```
st.number_input(label, min_value=None, max_value=None, value=, step=None,  
format=None, key=None, help=None, on_change=None, args=None, kwargs=None, *,  
disabled=False)
```

## Parameters

### **label** (*str*)

A short label explaining to the user what this input is for.

### **min\_value** (*int or float or None*)

The minimum permitted value. If None, there will be no minimum.

### **max\_value** (*int or float or None*)

The maximum permitted value. If None, there will be no maximum.

### **value** (*int or float or None*)

The value of this widget when it first renders. Defaults to min\_value, or 0.0 if min\_value is None

### **step** (*int or float or None*)

The stepping interval. Defaults to 1 if the value is an int, 0.01 otherwise. If the value is not specified, the format parameter will be used.

### **format** (*str or None*)

A printf-style format string controlling how the interface should display numbers. Output must be purely numeric. This does not impact the return value. Valid formatters: %d %e %f %g %i %u

### **key** (*str or int*)

An optional string or integer to use as the unique key for the widget. If this is omitted, a key will be generated for the widget based on its content. Multiple widgets of the same type may not share the same key.

### **help** (*str*)

An optional tooltip that gets displayed next to the input.

### **on\_change** (*callable*)

An optional callback invoked when this number\_input's value changes.

### **args** (*tuple*)

An optional tuple of args to pass to the callback.

### **kwargs** (*dict*)

An optional dict of kwargs to pass to the callback.

### **disabled** (*bool*)

An optional boolean, which disables the number input if set to True. The default is False. This argument can only be supplied by keyword.

## **Returns**

(*int or float*)

The current value of the numeric input widget. The return type will match the data type of the value parameter.

## Example

```
number = st.number_input('Insert a number')  
st.write('The current number is ', number)
```



Insert a number

- +

The current number is `0.0`

[\(view standalone Streamlit app\)](#)

## Was this page helpful?

Yes

No

[Suggest edits](#)



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← **Previous:** [st.text\\_input](#)

**Next:** [st.text\\_area](#) →

---

[Home](#)

[Contact Us](#)

[Community](#)



Copyright © 2022, Streamlit Inc.



# st.plotly\_chart

v1.9.0

Display an interactive Plotly chart.

Plotly is a charting library for Python. The arguments to this function closely follow the ones for Plotly's *plot()* function. You can find more about Plotly at <https://plot.ly/python>.

To show Plotly charts in Streamlit, call `st.plotly_chart` wherever you would call Plotly's *py.plot* or *py.iplot*.

## Function signature

```
st.plotly_chart(figure_or_data, use_container_width=False, sharing="streamlit",  
**kwargs)
```

## Parameters

**figure\_or\_data** (`plotly.graph_objs.Figure`, `plotly.graph_objs.Data`,)

dict/list of `plotly.graph_objs.Figure`/`Data`

See <https://plot.ly/python/> for examples of graph descriptions.

**use\_container\_width** (`bool`)

If True, set the chart width to the column width. This takes precedence over the figure's native *width* value.

**sharing** (`{'streamlit', 'private', 'secret', 'public'}`)

Use 'streamlit' to insert the plot and all its dependencies directly in the Streamlit app using plotly's offline mode (default). Use any other sharing mode to send the chart to Plotly chart studio, which requires an account. See <https://plotly.com/chart-studio/> for more information.

**\*\*kwargs (null)**

Any argument accepted by Plotly's *plot()* function.

## Example

The example below comes straight from the examples at <https://plot.ly/python>:

```
import streamlit as st
import plotly.figure_factory as ff
import numpy as np

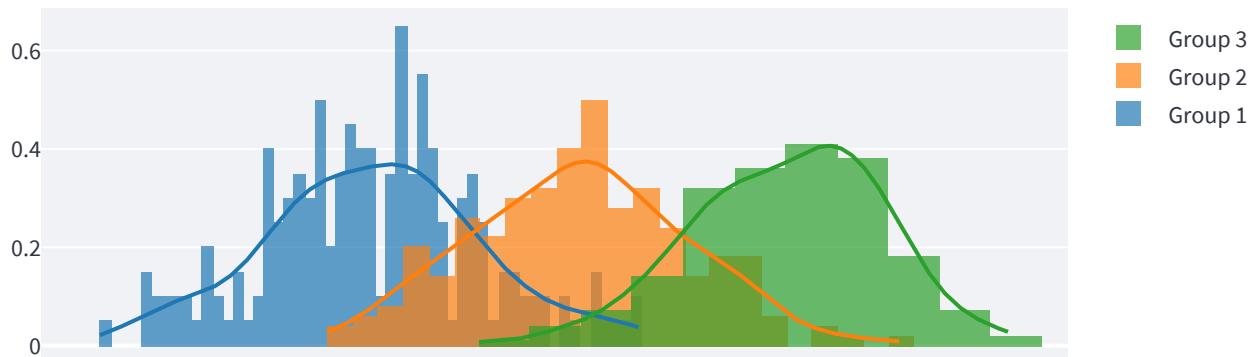
# Add histogram data
x1 = np.random.randn(200) - 2
x2 = np.random.randn(200)
x3 = np.random.randn(200) + 2

# Group data together
hist_data = [x1, x2, x3]

group_labels = ['Group 1', 'Group 2', 'Group 3']

# Create distplot with custom bin_size
fig = ff.create_distplot(
    hist_data, group_labels, bin_size=[.1, .25, .5])

# Plot!
st.plotly_chart(fig, use_container_width=True)
```



[\(view standalone Streamlit app\)](#)

## Was this page helpful?

Yes    No

[Suggest edits](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← Previous: [st.vega\\_lite\\_chart](#)

Next: [st.bokeh\\_chart](#) →





# st.progress

v1.9.0

Display a progress bar.

## Function signature

```
st.progress(value)
```

## Parameters

**value** (*int or float*)

$0 \leq \text{value} \leq 100$  for int

$0.0 \leq \text{value} \leq 1.0$  for float

## Example

Here is an example of a progress bar increasing over time:

```
import time

my_bar = st.progress(0)

for percent_complete in range(100):
    time.sleep(0.1)
    my_bar.progress(percent_complete + 1)
```



Was this page helpful?

Yes

No

 Suggest edits

---

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[← Previous: Status elements](#)

[Next: st.spinner →](#)

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# st.pydeck\_chart

v1.9.0

Draw a chart using the PyDeck library.

This supports 3D maps, point clouds, and more! More info about PyDeck at <https://deckgl.readthedocs.io/en/latest/>.

These docs are also quite useful:

- DeckGL docs: <https://github.com/uber/deck.gl/tree/master/docs>
- DeckGL JSON docs: <https://github.com/uber/deck.gl/tree/master/modules/json>

When using this command, we advise all users to use a personal Mapbox token. This ensures the map tiles used in this chart are more robust. You can do this with the `mapbox.token` config option.

To get a token for yourself, create an account at <https://mapbox.com>. It's free! (for moderate usage levels). For more info on how to set config options, see <https://docs.streamlit.io/library/advanced-features/configuration#set-configuration-options>

## Function signature

```
st.pydeck_chart(pydeck_obj=None, use_container_width=False)
```

## Parameters

**spec** (*pydeck.Deck or None*)

Object specifying the PyDeck chart to draw.

## Example

Here's a chart using a HexagonLayer and a ScatterplotLayer on top of the light map style:

```
df = pd.DataFrame(  
    np.random.randn(1000, 2) / [50, 50] + [37.76, -122.4],  
    columns=['lat', 'lon'])  
  
st.pydeck_chart(pdk.Deck(  
    map_style='mapbox://styles/mapbox/light-v9',  
    initial_view_state=pdk.ViewState(  
        latitude=37.76,  
        longitude=-122.4,  
        zoom=11,  
        pitch=50,  
    ),  
    layers=[  
        pdk.Layer(  
            'HexagonLayer',  
            data=df,  
            get_position='[lon, lat]',  
            radius=200,  
            elevation_scale=4,  
            elevation_range=[0, 1000],  
            pickable=True,  
            extruded=True,  
        ),  
        pdk.Layer(  
            'ScatterplotLayer',  
            data=df,  
            get_position='[lon, lat]',  
            get_color='[200, 30, 0, 160]',  
            get_radius=200,  
        ),  
    ],  
))
```

([view standalone Streamlit app](#)).

## Was this page helpful?

Yes      No

[Suggest edits](#)



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# st.pyplot

v1.9.0

Display a matplotlib.pyplot figure.

## Function signature

```
st.pyplot(fig=None, clear_figure=None, **kwargs)
```

## Parameters

### `fig (Matplotlib Figure)`

The figure to plot. When this argument isn't specified, this function will render the global figure (but this is deprecated, as described below)

### `clear_figure (bool)`

If True, the figure will be cleared after being rendered. If False, the figure will not be cleared after being rendered. If left unspecified, we pick a default based on the value of *fig*.

- If *fig* is set, defaults to *False*.
- If *fig* is not set, defaults to *True*. This simulates Jupyter's approach to matplotlib rendering.

### `**kwargs (any)`

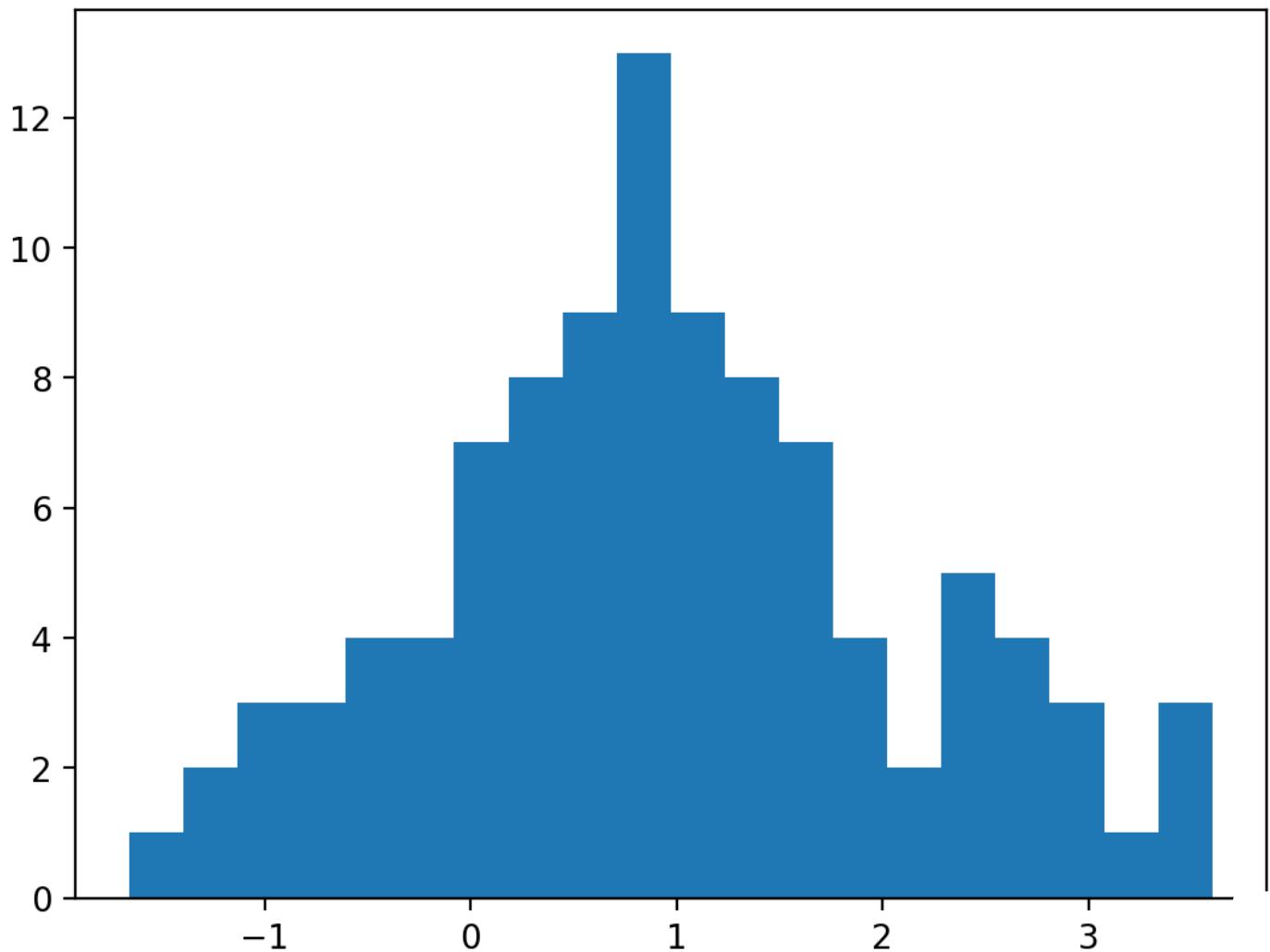
Arguments to pass to Matplotlib's savefig function.

## Example

```
import matplotlib.pyplot as plt
import numpy as np

arr = np.random.normal(1, 1, size=100)
fig, ax = plt.subplots()
ax.hist(arr, bins=20)

st.pyplot(fig)
```



[\(view standalone Streamlit app\)](#)

## Notes

### Note

Deprecation warning. After December 1st, 2020, we will remove the ability to specify no arguments in `st.pyplot()`, as that requires the use of Matplotlib's global figure object, which is not thread-safe. So please

always pass a figure object as shown in the example section above.

Matplotlib support several different types of "backends". If you're getting an error using Matplotlib with Streamlit, try setting your backend to "TkAgg":

```
echo "backend: TkAgg" >> ~/.matplotlib/matplotlibrc
```



For more information, see [https://matplotlib.org/faq/usage\\_faq.html](https://matplotlib.org/faq/usage_faq.html).

## Was this page helpful?

Yes    No

[Suggest edits](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[← Previous: st.bar\\_chart](#)

[Next: st.altair\\_chart →](#)

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# st.radio

v1.9.0

Display a radio button widget.

## Function signature

```
st.radio(label, options, index=0, format_func=special_internal_function,  
key=None, help=None, on_change=None, args=None, kwargs=None, *, disabled=False)
```

## Parameters

### **label** (*str*)

A short label explaining to the user what this radio group is for.

### **options** (*Sequence, numpy.ndarray, pandas.Series, pandas.DataFrame, or pandas.Index*)

Labels for the radio options. This will be cast to str internally by default. For pandas.DataFrame, the first column is selected.

### **index** (*int*)

The index of the preselected option on first render.

### **format\_func** (*function*)

Function to modify the display of radio options. It receives the raw option as an argument and should output the label to be shown for that option. This has no impact on the return value of the radio.

## **key** (*str or int*)

An optional string or integer to use as the unique key for the widget. If this is omitted, a key will be generated for the widget based on its content. Multiple widgets of the same type may not share the same key.

## **help** (*str*)

An optional tooltip that gets displayed next to the radio.

## **on\_change** (*callable*)

An optional callback invoked when this radio's value changes.

## **args** (*tuple*)

An optional tuple of args to pass to the callback.

## **kwargs** (*dict*)

An optional dict of kwargs to pass to the callback.

## **disabled** (*bool*)

An optional boolean, which disables the radio button if set to True. The default is False. This argument can only be supplied by keyword.

## **Returns**

### **(any)**

The selected option.

## **Example**

```
genre = st.radio(  
    "What's your favorite movie genre",  
    ('Comedy', 'Drama', 'Documentary'))
```

```
if genre == 'Comedy':  
    st.write('You selected comedy.')  
else:  
    st.write("You didn't select comedy.")
```



What's your favorite movie genre

Comedy

Drama

Documentary

You selected comedy.

[\(view standalone Streamlit app\)](#)

## Featured videos

Check out our video on how to use one of Streamlit's core functions, the radio button!

Streamlit Shorts: How to make a radio button

Copy link



Watch on YouTube

In the video below, we'll take it a step further and learn how to combine a button, checkbox and radio button!



Watch on  YouTube

## Was this page helpful?

 Yes     No

 [Suggest edits](#)



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: st.checkbox](#)

[Next: st.selectbox](#) →



Copyright © 2022, Streamlit Inc.



## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*remove*
    - [st.button](#)
    - [st.download\\_button](#)
    - [st.checkbox](#)
    - [st.radio](#)
    - [st.selectbox](#)
    - [st.multiselect](#)
    - [st.slider](#)
    - [st.select\\_slider](#)
    - [st.text\\_input](#)
    - [st.number\\_input](#)
    - [st.text\\_area](#)
    - [st.date\\_input](#)
    - [st.time\\_input](#)
    - [st.file\\_uploader](#)
    - [st.camera\\_input](#)
    - [st.color\\_picker](#)
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*add*
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)

- [Cheat sheet](#)

- [cloud](#)

## [Streamlit Cloud](#)

- [Get started](#)
- [add](#)
- [Trust and Security](#)
- [Release notes](#) [open in new](#)
- [Troubleshooting](#)

- [school](#)

## [Knowledge base](#)

- [Tutorials](#)
- [add](#)
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home/](#)
- [Streamlit library/](#)
- [API reference/](#)
- [Input widgets/](#)
- [st.select\\_slider](#)

## st.select\_slider

8

Streamlit Version [v1.9.0](#) 

Display a slider widget to select items from a list.

This also allows you to render a range slider by passing a two-element tuple or list as the *value*.

The difference between *st.select\_slider* and *st.slider* is that *select\_slider* accepts any datatype and takes an iterable set of options, while *slider* only accepts numerical or date/time data and takes a range as input.

### Function signature

```
st.select_slider(label, options=[], value=None, format_func=special_internal_function, key=None, help=None,  
                 on_change=None, args=None, kwargs=None, *, disabled=False)
```

#### Parameters

label (str)

A short label explaining to the user what this slider is for.

options (Sequence,  
numpy.ndarray,  
pandas.Series,  
pandas.DataFrame, or  
pandas.Index)

Labels for the slider options. All options will be cast to str internally by default. For pandas.DataFrame, the first column is selected.

value (a supported type or a  
tuple/list of supported types  
or None)

The value of the slider when it first renders. If a tuple/list of two values is passed here, then a range slider with those lower and upper bounds is rendered. For example, if set to (1, 10) the slider will have a selectable range between 1 and 10. Defaults to first option.

format_func (function)	Function to modify the display of the labels from the options. argument. It receives the option as an argument and its output will be cast to str.
key (str or int)	An optional string or integer to use as the unique key for the widget. If this is omitted, a key will be generated for the widget based on its content. Multiple widgets of the same type may not share the same key.
help (str)	An optional tooltip that gets displayed next to the select slider.
on_change (callable)	An optional callback invoked when this select_slider's value changes.
args (tuple)	An optional tuple of args to pass to the callback.
kwargs (dict)	An optional dict of kwargs to pass to the callback.
disabled (bool)	An optional boolean, which disables the select slider if set to True. The default is False. This argument can only be supplied by keyword.

## Returns

(any value or tuple of any value)	The current value of the slider widget. The return type will match the data type of the value parameter.
-----------------------------------	--

## Examples

```
color = st.select_slider(
    'Select a color of the rainbow',
    options=['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet'])
st.write('My favorite color is', color)
```

And here's an example of a range select slider:

```
start_color, end_color = st.select_slider(
    'Select a range of color wavelength',
    options=['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet'],
    value=('red', 'blue'))
st.write('You selected wavelengths between', start_color, 'and', end_color)
```

Select a color of the rainbow

red

red

violet

My favorite color is red

Select a range of color wavelength

red

blue

red

violet

You selected wavelengths between red and blue

[\(view standalone Streamlit app\)](#)

## Featured videos

8

Check out our video on how to use one of Streamlit's core functions, the select slider! 

Streamlit Shorts: Ho...



In the video below, we'll take it a step further and make a double-ended slider.

Streamlit shorts: Ho...



Was this page helpful?

Yes  No

[edit](#) [Suggest edits](#)

[forum](#)

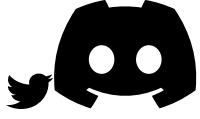
Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

←[Previous: st.slider](#)[Next: st.text\\_input](#)→

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*remove*
    - [st.button](#)
    - [st.download\\_button](#)
    - [st.checkbox](#)
    - [st.radio](#)
    - [st.selectbox](#)
    - [st.multiselect](#)
    - [st.slider](#)
    - [st.select\\_slider](#)
    - [st.text\\_input](#)
    - [st.number\\_input](#)
    - [st.text\\_area](#)
    - [st.date\\_input](#)
    - [st.time\\_input](#)
    - [st.file\\_uploader](#)
    - [st.camera\\_input](#)
    - [st.color\\_picker](#)
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*add*
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)

- [Cheat sheet](#)

- [cloud](#)

## [Streamlit Cloud](#)

- [Get started](#)
- [add](#)
- [Trust and Security](#)
- [Release notes](#) [open in new](#)
- [Troubleshooting](#)

- [school](#)

## [Knowledge base](#)

- [Tutorials](#)
- [add](#)
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home](#)/
- [Streamlit library](#)/
- [API reference](#)/
- [Input widgets](#)/
- [st.selectbox](#)

## st.selectbox

8

Streamlit Version [v1.9.0](#) 

Display a select widget.

### Function signature

```
st.selectbox(label, options, index=0, format_func=special_internal_function, key=None, help=None,
            on_change=None, args=None, kwargs=None, *, disabled=False)
```

#### Parameters

label (str) A short label explaining to the user what this select widget is for.

options (Sequence, numpy.ndarray, pandas.Series, pandas.DataFrame, or pandas.Index) Labels for the select options. This will be cast to str internally by default. For pandas.DataFrame, the first column is selected.

index (int) The index of the preselected option on first render.

format\_func (function) Function to modify the display of the labels. It receives the option as an argument and its output will be cast to str.

key (str or int) An optional string or integer to use as the unique key for the widget. If this is omitted, a key will be generated for the widget based on its content. Multiple widgets of the same type may not share the same key.

help (str) An optional tooltip that gets displayed next to the selectbox.

on\_change (callable) An optional callback invoked when this selectbox's value changes.

args (tuple) An optional tuple of args to pass to the callback.

kwargs (dict) An optional dict of kwargs to pass to the callback.

disabled (bool) An optional boolean, which disables the selectbox if set to True. The default is False. This argument can only be supplied by keyword.

Returns

(any) The selected option

## Example

```
option = st.selectbox(  
    'How would you like to be contacted?',  
    ('Email', 'Home phone', 'Mobile phone'))  
  
st.write('You selected:', option)
```



Please wait...

[\(view standalone Streamlit app\)](#)

Was this page helpful?

Yes No

[edit](#)[Suggest edits](#)

[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

←[Previous: st.radio](#)[Next: st.multiselect](#)→

---

[Home](#)[Contact Us](#)[Community](#)





## Documentation

*search*

Search

- *description*

### Streamlit library

- Get started  
*add*
- API reference  
*remove*
  - Write and magic  
*add*
  - Text elements  
*add*
  - Data display elements  
*add*
  - Chart elements  
*add*
  - Input widgets  
*add*
  - Media elements  
*add*
  - Layouts and containers  
*add*
  - Status elements  
*add*
  - Control flow  
*add*
  - Utilities  
*remove*
    - st.set\_page\_config
    - st.echo
    - st.help
    - st.experimental\_show
    - st.experimental\_get\_query\_params
    - st.experimental\_set\_query\_params
  - Mutate charts
  - State management
  - Performance  
*add*
- Advanced features  
*add*
- Components  
*add*
- Changelog
- Cheat sheet

- *cloud*

### Streamlit Cloud

- Get started  
*add*
- Trust and Security
- Release notes*open in new*
- Troubleshooting

- *school*

## [Knowledge base](#)

- [Tutorials](#)  
*add*
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home/](#)
- [Streamlit library/](#)
- [API reference/](#)
- [Utilities/](#)
- [st.set\\_page\\_config](#)

## st.set\_page\_config

8

Streamlit Version [v1.9.0](#) ▾

Configures the default settings of the page.

### Note

This must be the first Streamlit command used in your app, and must only be set once.

### Function signature

```
st.set_page_config(page_title=None, page_icon=None, layout="centered", initial_sidebar_state="auto",  
                   menu_items=None)
```

### Parameters

page\_title (str or None) The page title, shown in the browser tab. If None, defaults to the filename of the script ("app.py" would show "app • Streamlit").

page\_icon (Anything supported by st.image or str or None) The page favicon. Besides the types supported by *st.image* (like URLs or numpy arrays), you can pass in an emoji as a string ("") or a shortcode ("":shark:""). If you're feeling lucky, try "random" for a random emoji! Emoji icons are courtesy of Twemoji and loaded from MaxCDN.

layout ("centered" or "wide") How the page content should be laid out. Defaults to "centered", which constrains the elements into a centered column of fixed width; "wide" uses the entire screen.

initial\_sidebar\_state ("auto" or "expanded" or "collapsed") How the sidebar should start out. Defaults to "auto", which hides the sidebar on mobile-sized devices, and shows it otherwise. "expanded" shows the sidebar initially; "collapsed" hides it.

menu\_items (dict) Configure the menu that appears on the top-right side of this app. The keys in this dict denote the menu item you'd like to configure:

- "Get help": str or None  
The URL this menu item should point to. If None, hides this menu item.
- "Report a Bug": str or None

The URL this menu item should point to. If None, hides this menu item.

- "About": str or None

A markdown string to show in the About dialog. If None, only shows Streamlit's default About text.

## Example

```
st.set_page_config(  
    page_title="Ex-stream-ly Cool App",  
    page_icon="💡",  
    layout="wide",  
    initial_sidebar_state="expanded",  
    menu_items={  
        'Get Help': 'https://www.extremelycoolapp.com/help',  
        'Report a bug': "https://www.extremelycoolapp.com/bug",  
        'About': "# This is a header. This is an *extremely* cool app!"  
    }  
)
```

Was this page helpful?

Yes  No

[edit](#)[Suggest edits](#)

[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: Utilities](#) [Next: st.echo](#) →

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*add*
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*remove*
    - [st.sidebar](#)
    - [st.columns](#)
    - [st.expander](#)
    - [st.container](#)
    - [st.empty](#)
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [\*cloud\*](#)

### [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notesopen in new](#)
- [Troubleshooting](#)

- [\*school\*](#)

## Knowledge base

- [Tutorials](#)  
*add*
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home/](#)
- [Streamlit library/](#)
- [API reference/](#)
- [Layouts and containers/](#)
- [st.sidebar](#)

## st.sidebar

8

### Add widgets to sidebar

8

Not only can you add interactivity to your app with widgets, you can organize them into a sidebar. Elements can be passed to `st.sidebar` using object notation and `with` notation.

The following two snippets are equivalent:

```
# Object notation
st.sidebar.[element_name]

# "with" notation
with st.sidebar:
    st.[element_name]
```

Each element that's passed to `st.sidebar` is pinned to the left, allowing users to focus on the content in your app.

Here's an example of how you'd add a selectbox and a radio button to your sidebar:

```
import streamlit as st

# Using object notation
add_selectbox = st.sidebar.selectbox(
    "How would you like to be contacted?",
    ("Email", "Home phone", "Mobile phone")
)

# Using "with" notation
with st.sidebar:
    add_radio = st.radio(
        "Choose a shipping method",
        ("Standard (5-15 days)", "Express (2-5 days)")
)
```

*priority\_high*

### Important

The only elements that aren't supported using object notation are `st.echo` and `st.spinner`. To use these elements, you must use `with` notation.

Here's an example of how you'd add `st.echo` and `st.spinner` to your sidebar:

```
import streamlit as st

with st.sidebar:
    with st.echo():
        pass
```

```
st.write("This code will be printed to the sidebar.")
```

```
with st.spinner("Loading..."):  
    time.sleep(5)  
st.success("Done!")
```

Was this page helpful?

Yes  No

[edit](#)[Suggest edits](#)

[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: Layouts and containers](#) [Next: st.columns](#) →

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.



# st.slider

v1.9.0

Display a slider widget.

This supports int, float, date, time, and datetime types.

This also allows you to render a range slider by passing a two-element tuple or list as the *value*.

The difference between *st.slider* and *st.select\_slider* is that *slider* only accepts numerical or date/time data and takes a range as input, while *select\_slider* accepts any datatype and takes an iterable set of options.

## Function signature

```
st.slider(label, min_value=None, max_value=None, value=None, step=None,  
format=None, key=None, help=None, on_change=None, args=None, kwargs=None, *,  
disabled=False)
```

## Parameters

### label (str)

A short label explaining to the user what this slider is for.

### min\_value (a supported type or None)

The minimum permitted value. Defaults to 0 if the value is an int, 0.0 if a float, value - timedelta(days=14) if a date/datetime, time.min if a time

### max\_value (a supported type or None)

The maximum permitted value. Defaults to 100 if the value is an int, 1.0 if a float, value + timedelta(days=14) if a date/datetime, time.max if a time

## **value** (*a supported type or a tuple/list of supported types or None*)

The value of the slider when it first renders. If a tuple/list of two values is passed here, then a range slider with those lower and upper bounds is rendered. For example, if set to `(1, 10)` the slider will have a selectable range between 1 and 10. Defaults to `min_value`.

## **step** (*int/float/timedelta or None*)

The stepping interval. Defaults to 1 if the value is an int, 0.01 if a float, timedelta(days=1) if a date/datetime, timedelta(minutes=15) if a time (or if `max_value - min_value < 1 day`)

## **format** (*str or None*)

A printf-style format string controlling how the interface should display numbers. This does not impact the return value. Formatter for int/float supports: `%d %e %f %g %i` Formatter for date/time/datetime uses Moment.js notation: <https://momentjs.com/docs/#/displaying/format/>

## **key** (*str or int*)

An optional string or integer to use as the unique key for the widget. If this is omitted, a key will be generated for the widget based on its content. Multiple widgets of the same type may not share the same key.

## **help** (*str*)

An optional tooltip that gets displayed next to the slider.

## **on\_change** (*callable*)

An optional callback invoked when this slider's value changes.

## **args** (*tuple*)

An optional tuple of args to pass to the callback.

## **kwargs** (*dict*)

An optional dict of kwargs to pass to the callback.

## disabled (bool)

An optional boolean, which disables the slider if set to True. The default is False. This argument can only be supplied by keyword.

### Returns

(int/float/date/time/datetime or tuple of int/float/date/time/datetime)

The current value of the slider widget. The return type will match the data type of the value parameter.

## Examples

```
age = st.slider('How old are you?', 0, 130, 25)
st.write("I'm ", age, 'years old')
```

And here's an example of a range slider:

```
values = st.slider(
    'Select a range of values',
    0.0, 100.0, (25.0, 75.0))
st.write('Values:', values)
```

This is a range time slider:

```
from datetime import time
appointment = st.slider(
    "Schedule your appointment:",
    value=(time(11, 30), time(12, 45)))
st.write("You're scheduled for:", appointment)
```

Finally, a datetime slider:

```
from datetime import datetime
start_time = st.slider(
    "When do you start?",
    value=datetime(2020, 1, 1, 9, 30),
    format="%MM/DD/YY - hh:mm")
st.write("Start time:", start_time)
```

# Slider

Loading...

I'm [25](#) years old.

[Dance slider](#)

(view standalone Streamlit app)

## Featured videos

Check out our video on how to use one of Streamlit's core functions, the slider!

Streamlit Shorts: How to make a slider



[Copy link](#)



In the video below, we'll take it a step further and make a double-ended slider.



## Was this page helpful?

 Yes     No

 [Suggest edits](#)

---

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[← Previous: st.multiselect](#)

[Next: st.select\\_slider →](#)



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*add*
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*add*
  - [Status elements](#)  
*remove*
    - [st.progress](#)
    - [st.spinner](#)
    - [st.balloons](#)
    - [st.snow](#)
    - [st.error](#)
    - [st.warning](#)
    - [st.info](#)
    - [st.success](#)
    - [st.exception](#)
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [\*cloud\*](#)

### [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)

- [Release notes](#)<sub>open in new</sub>
- [Troubleshooting](#)

- [school](#)

## [Knowledge base](#)

- [Tutorials](#)  
*add*
  - [Using Streamlit](#)
  - [Streamlit Components](#)
  - [Installing dependencies](#)
  - [Deployment issues](#)
- [Home](#)/
  - [Streamlit library](#)/
  - [API reference](#)/
  - [Status elements](#)/
  - [st.spinner](#)

## st.spinner

8

Streamlit Version [v1.9.0](#) 

Temporarily displays a message while executing a block of code.

### Function signature

```
st.spinner(text="In progress...")
```

#### Parameters

text (str) A message to display while executing that block

#### Example

```
with st.spinner('Wait for it...'):  
    time.sleep(5)  
st.success('Done!')
```

Was this page helpful?

[thumb\\_up](#) Yes [thumb\\_down](#) No  
[edit](#)[Suggest edits](#)  
[forum](#)

#### Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

←[Previous: st.progress](#)[Next: st.balloons](#)→

---

[Home](#)[Contact Us](#)[Community](#)





## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*add*
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*add*
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*remove*
    - [st.stop](#)
    - [st.form](#)
    - [st.form\\_submit\\_button](#)
    - [st.experimental\\_rerun](#)
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [cloud](#)

### [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notesopen in new](#)
- [Troubleshooting](#)

- [school](#)

### [Knowledge base](#)

- [Tutorials](#)
- [add](#)
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home/](#)
- [Streamlit library/](#)
- [API reference/](#)
- [Control flow/](#)
- [st.stop](#)

## st.stop

Streamlit Version [v1.9.0](#) ▾

Stops execution immediately.

Streamlit will not run any statements after `st.stop()`. We recommend rendering a message to explain why the script has stopped. When run outside of Streamlit, this will raise an Exception.

### Function signature

#### st.stop()

### Example

```
name = st.text_input('Name')
if not name:
    st.warning('Please input a name.')
    st.stop()
st.success('Thank you for inputting a name.')
```

### Was this page helpful?

[\*thumb\\_up\* Yes](#)      [\*thumb\\_down\* No](#)

[edit](#) [Suggest edits](#)

Still have questions?

*forum*

Our [forums](#) are full of helpful information and Streamlit experts.

[\*\*← Previous: Control flow\*\*](#)

[\*\*Next: st.form →\*\*](#)

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.



# st.subheader

v1.9.0

Display text in subheader formatting.

## Function signature

```
st.subheader(body, anchor=None)
```

## Parameters

### body (str)

The text to display.

### anchor (str)

The anchor name of the header that can be accessed with #anchor in the URL. If omitted, it generates an anchor using the body.

## Example

```
st.subheader('This is a subheader')
```

This is a subheader

Was this page helpful?

Yes

No



Suggest edits

---

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

← **Previous:** st.header

**Next:** st.caption →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# st.success

v1.9.0

Display a success message.

## Function signature

```
st.success(body)
```

## Parameters

**body (str)**

The success text to display.

## Example

```
st.success('This is a success message!')
```



## Was this page helpful?

Yes    No

[Suggest edits](#)



# Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

← **Previous:** [st.info](#)

**Next:** [st.exception](#) →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*remove*
    - [st.dataframe](#)
    - [st.table](#)
    - [st.metric](#)
    - [st.json](#)
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*add*
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*add*
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [\*cloud\*](#)

### [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notesopen in new](#)
- [Troubleshooting](#)

- [\*school\*](#)

### [Knowledge base](#)

- [Tutorials](#)
- *add*
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

[Home](#) / [Streamlit library](#) / [API reference](#) / [Data display elements](#) / [\*\*st.table\*\*](#)

# st.table

8

Streamlit Version

**v1.9.0**

Display a static table.

This differs from `st.dataframe` in that the table in this case is static: its entire contents are laid out directly on the page.

## Function signature

`st.table(data=None)`

### Parameters

<code>data</code> (pandas.DataFrame, pandas.Styler, pyarrow.Table, numpy.ndarray, Iterable, dict, or None)	The table data. Pyarrow tables are not supported by Streamlit's legacy DataFrame serialization (i.e. with <code>config.dataFrameSerialization = "legacy"</code> ). To use pyarrow tables, please enable pyarrow by changing the config setting, <code>config.dataFrameSerialization = "arrow"</code> .
--	--

### Example

```
df = pd.DataFrame(  
    np.random.randn(10, 5),  
    columns=('col %d' % i for i in range(5)))  
  
st.table(df)
```

Please wait...

Made with Streamlit

([view standalone Streamlit app](#)).

Hosted with Streamlit

Was this page helpful?

Yes No  
[edit](#)[Suggest edits](#)  
[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

←[Previous: st.dataframe](#)[Next: st.metric](#)→

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.



# st.text

v1.9.0

Write fixed-width and preformatted text.

## Function signature

```
st.text(body)
```

## Parameters

**body** (*str*)

The string to display.

## Example

```
st.text('This is some text.')
```



This is some text.

## Was this page helpful?

Yes

No

[Suggest edits](#)



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[\*\*← Previous:\*\* st.code](#)

**Next:** st.latex [\*\*→\*\*](#)

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# st.text\_area

v1.9.0

Display a multi-line text input widget.

## Function signature

```
st.text_area(label, value="", height=None, max_chars=None, key=None, help=None, on_change=None, args=None, kwargs=None, *, placeholder=None, disabled=False)
```

## Parameters

### **label** (*str*)

A short label explaining to the user what this input is for.

### **value** (*any*)

The text value of this widget when it first renders. This will be cast to str internally.

### **height** (*int or None*)

Desired height of the UI element expressed in pixels. If None, a default height is used.

### **max\_chars** (*int or None*)

Maximum number of characters allowed in text area.

### **key** (*str or int*)

An optional string or integer to use as the unique key for the widget. If this is omitted, a key will be generated

for the widget based on its content. Multiple widgets of the same type may not share the same key.

### **help (str)**

An optional tooltip that gets displayed next to the textarea.

### **on\_change (callable)**

An optional callback invoked when this text\_area's value changes.

### **args (tuple)**

An optional tuple of args to pass to the callback.

### **kwargs (dict)**

An optional dict of kwargs to pass to the callback.

### **placeholder (str or None)**

An optional string displayed when the text area is empty. If None, no text is displayed. This argument can only be supplied by keyword.

### **disabled (bool)**

An optional boolean, which disables the text area if set to True. The default is False. This argument can only be supplied by keyword.

## **Returns**

### **(str)**

The current value of the text input widget.

## **Example**

```
txt = st.text_area('Text to analyze', '')
```

```
It was the best of times, it was the worst of times, it was  
the age of wisdom, it was the age of foolishness, it was
```

the epoch of belief, it was the epoch of incredulity, it  
was the season of Light, it was the season of Darkness, it  
was the spring of hope, it was the winter of despair, (...)  
'''

```
st.write('Sentiment:', run_sentiment_analysis(txt))
```

## Was this page helpful?

 Yes       No

 [Suggest edits](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

[← Previous: st.number\\_input](#)

[Next: st.date\\_input →](#)

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# st.text\_input

v1.9.0

Display a single-line text input widget.

## Function signature

```
st.text_input(label, value="", max_chars=None, key=None, type="default",
help=None, autocomplete=None, on_change=None, args=None, kwargs=None, *,
placeholder=None, disabled=False)
```

## Parameters

### label (str)

A short label explaining to the user what this input is for.

### value (any)

The text value of this widget when it first renders. This will be cast to str internally.

### max\_chars (int or None)

Max number of characters allowed in text input.

### key (str or int)

An optional string or integer to use as the unique key for the widget. If this is omitted, a key will be generated for the widget based on its content. Multiple widgets of the same type may not share the same key.

### type (str)

The type of the text input. This can be either "default" (for a regular text input), or "password" (for a text input that masks the user's typed value). Defaults to "default".

### **help (str)**

An optional tooltip that gets displayed next to the input.

### **autocomplete (str)**

An optional value that will be passed to the <input> element's autocomplete property. If unspecified, this value will be set to "new-password" for "password" inputs, and the empty string for "default" inputs. For more details, see <https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes/autocomplete>

### **on\_change (callable)**

An optional callback invoked when this text\_input's value changes.

### **args (tuple)**

An optional tuple of args to pass to the callback.

### **kwargs (dict)**

An optional dict of kwargs to pass to the callback.

### **placeholder (str or None)**

An optional string displayed when the text input is empty. If None, no text is displayed. This argument can only be supplied by keyword.

### **disabled (bool)**

An optional boolean, which disables the text input if set to True. The default is False. This argument can only be supplied by keyword.

## **Returns**

(str)

The current value of the text input widget.



## Example

```
title = st.text_input('Movie title', 'Life of Brian')
st.write('The current movie title is', title)
```

Movie title

Life of Brian

The current movie title is Life of Brian

[\(view standalone Streamlit app\)](#)

## Was this page helpful?

Yes    No

[Suggest edits](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← Previous: [st.select\\_slider](#)

Next: [st.number\\_input](#) →



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*remove*
    - [st.button](#)
    - [st.download\\_button](#)
    - [st.checkbox](#)
    - [st.radio](#)
    - [st.selectbox](#)
    - [st.multiselect](#)
    - [st.slider](#)
    - [st.select\\_slider](#)
    - [st.text\\_input](#)
    - [st.number\\_input](#)
    - [st.text\\_area](#)
    - [st.date\\_input](#)
    - [st.time\\_input](#)
    - [st.file\\_uploader](#)
    - [st.camera\\_input](#)
    - [st.color\\_picker](#)
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*add*
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)

- [Cheat sheet](#)

- [\*cloud\*](#)

## [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notes](#)*open in new*
- [Troubleshooting](#)

- [\*school\*](#)

## [Knowledge base](#)

- [Tutorials](#)  
*add*
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

[Home](#) / [Streamlit library](#) / [API reference](#) / [Input widgets](#) / [\*\*st.time\\_input\*\*](#)

# **st.time\_input**

Streamlit Version

**v1.9.0**

Display a time input widget.

## Function signature

```
st.time_input(label, value=None, key=None, help=None, on_change=None, args=None, kwargs=None, *,  
disabled=False)
```

## Parameters

label (str)

A short label explaining to the user what this time input is for.

value (datetime.time/datetime.datetime)

The value of this widget when it first renders. This will be cast to str internally. Defaults to the current time.

`key (str or int)`

An optional string or integer to use as the unique key for the widget. If this is omitted, a key will be generated for the widget based on its content. Multiple widgets of the same type may not share the same key.

`help (str)`

An optional tooltip that gets displayed next to the input.

`on_change (callable)`

An optional callback invoked when this `time_input`'s value changes.

`args (tuple)`

An optional tuple of args to pass to the callback.

`kwargs (dict)`

An optional dict of kwargs to pass to the callback.

`disabled (bool)`

An optional boolean, which disables the time input if set to True. The default is False. This argument can only be supplied by keyword.

## Returns

(datetime.time)

The current value of the time input widget.



## Example

```
t = st.time_input('Set an alarm for', datetime.time(8, 45))  
st.write('Alarm is set for', t)
```

Loading...

Alarm is set for 08:45:00

[\(view standalone Streamlit app\)](#)

## Was this page helpful?

*thumb\_up* Yes      *thumb\_down* No

*edit* [Suggest edits](#)

## Still have questions?

*forum*

Our [forums](#) are full of helpful information and Streamlit experts.

**Next:** [st.file\\_uploader](#) →

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.



# st.title

v1.9.0

Display text in title formatting.

Each document should have a single `st.title()`, although this is not enforced.

## Function signature

```
st.title(body, anchor=None)
```

## Parameters

### body (str)

The text to display.

### anchor (str)

The anchor name of the header that can be accessed with `#anchor` in the URL. If omitted, it generates an anchor using the body.

## Example

```
st.title('This is a title')
```



# This is a title

Was this page helpful?

Yes

No

 Suggest edits

---

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

← Previous: st.markdown

Next: st.header →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*remove*
    - [st.line\\_chart](#)
    - [st.area\\_chart](#)
    - [st.bar\\_chart](#)
    - [st.pyplot](#)
    - [st.altair\\_chart](#)
    - [st.vega\\_lite\\_chart](#)
    - [st.plotly\\_chart](#)
    - [st.bokeh\\_chart](#)
    - [st.pydeck\\_chart](#)
    - [st.graphviz\\_chart](#)
    - [st.map](#)
  - [Input widgets](#)  
*add*
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*add*
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [\*cloud\*](#)

### [Streamlit Cloud](#)

- [Get started](#)

- add*
- o [Trust and Security](#)
- o [Release notes](#) open in new
- o [Troubleshooting](#)
- [school](#)

## [Knowledge base](#)

- o [Tutorials](#)
- add*
- o [Using Streamlit](#)
- o [Streamlit Components](#)
- o [Installing dependencies](#)
- o [Deployment issues](#)

- [Home](#)/
- [Streamlit library](#)/
- [API reference](#)/
- [Chart elements](#)/
- [st.vega\\_lite\\_chart](#)

## st.vega\_lite\_chart

8

Streamlit Version v1.9.0 ▾

Display a chart using the Vega-Lite library.

### Function signature

```
st.vega_lite_chart(data=None, spec=None, use_container_width=False, **kwargs)
```

#### Parameters

data (pandas.DataFrame, pandas.Styler, pyarrow.Table, numpy.ndarray, Iterable, dict, or None)	Either the data to be plotted or a Vega-Lite spec containing the data (which more closely follows the Vega-Lite API). Pyarrow tables are not supported by Streamlit's legacy DataFrame serialization (i.e. with <code>config.dataFrameSerialization = "legacy"</code> ). To use pyarrow tables, please enable pyarrow by changing the config setting, <code>config.dataFrameSerialization = "arrow"</code> .
spec (dict or None)	The Vega-Lite spec for the chart. If the spec was already passed in the previous argument, this must be set to None. See <a href="https://vega.github.io/vega-lite/docs/">https://vega.github.io/vega-lite/docs/</a> for more info.
use_container_width (bool)	If True, set the chart width to the column width. This takes precedence over Vega-Lite's native <code>width</code> value.
**kwargs (any)	Same as spec, but as keywords.

## Example

```
import pandas as pd
import numpy as np

df = pd.DataFrame(
    np.random.randn(200, 3),
    columns=['a', 'b', 'c'])

st.vega_lite_chart(df, {
```

```
'mark': {'type': 'circle', 'tooltip': True},  
'encoding': {  
    'x': {'field': 'a', 'type': 'quantitative'},  
    'y': {'field': 'b', 'type': 'quantitative'},  
    'size': {'field': 'c', 'type': 'quantitative'},  
    'color': {'field': 'c', 'type': 'quantitative'}  
},  
})
```

[\(view standalone Streamlit app\)](#)

Examples of Vega-Lite usage without Streamlit can be found at <https://vega.github.io/vega-lite/examples/>. Most of those can be easily translated to the syntax shown above.

Was this page helpful?

Yes    No  
[edit](#)[Suggest edits](#)  
[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

←[Previous: st.altair\\_chart](#)[Next: st.plotly\\_chart](#)→

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.



# st.video

v1.9.0

Display a video player.

## Function signature

```
st.video(data, format="video/mp4", start_time=0)
```

## Parameters

**data** (*str, bytes, BytesIO, numpy.ndarray, or file opened with*)

io.open(). Raw video data, filename, or URL pointing to a video to load. Includes support for YouTube URLs. Numpy arrays and raw data formats must include all necessary file headers to match specified file format.

**format** (*str*)

The mime type for the video file. Defaults to 'video/mp4'. See <https://tools.ietf.org/html/rfc4281> for more info.

**start\_time** (*int*)

The time from which this element should start playing.

## Example

```
video_file = open('myvideo.mp4', 'rb')
video_bytes = video_file.read()

st.video(video_bytes)
```

## Video credit:

Creator: User *fxxu* from *Pixabay*.

License: Free for commercial use. No attribution required. <https://pixabay.com/en/service/license/>

URL: <https://pixabay.com/en/videos/star-long-exposure-starry-sky-6962/>

[\(view standalone Streamlit app\)](#)

### Note

Some videos may not display if they are encoded using MP4V (which is an export option in OpenCV), as this codec is not widely supported by browsers. Converting your video to H.264 will allow the video to be displayed in Streamlit. See this [StackOverflow post](#) or this [Streamlit forum post](#) for more information.

### Was this page helpful?

Yes

No

---

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

← [Previous: st.audio](#)

[Next: Layouts and containers](#) →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*add*
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*add*
  - [Status elements](#)  
*remove*
    - [st.progress](#)
    - [st.spinner](#)
    - [st.balloons](#)
    - [st.snow](#)
    - [st.error](#)
    - [st.warning](#)
    - [st.info](#)
    - [st.success](#)
    - [st.exception](#)
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [\*cloud\*](#)

### [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)

- [Release notes](#)<sub>open in new</sub>
- [Troubleshooting](#)

- [school](#)

## [Knowledge base](#)

- [Tutorials](#)  
*add*
  - [Using Streamlit](#)
  - [Streamlit Components](#)
  - [Installing dependencies](#)
  - [Deployment issues](#)
- [Home](#)/
  - [Streamlit library](#)/
  - [API reference](#)/
  - [Status elements](#)/
  - [st.warning](#)

## st.warning

8

Streamlit Version [v1.9.0](#) 

Display warning message.

### Function signature

`st.warning(body)`

#### Parameters

body (str) The warning text to display.

### Example

```
st.warning('This is a warning')
```

Was this page helpful?

Yes No

[edit](#)[Suggest edits](#)

[forum](#)

### Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

←[Previous: st.error](#)[Next: st.info](#)→

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.

## Documentation

*search*

Search

- *description*

### Streamlit library

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*remove*
    - [st.write](#)
    - [magic](#)
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*add*
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*add*
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- *cloud*

### Streamlit Cloud

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notes](#)*open in new*
- [Troubleshooting](#)

- *school*

### Knowledge base

- [Tutorials](#)

- [add](#)
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home/](#)
- [Streamlit library/](#)
- [API reference/](#)
- [Write and magic/](#)
- [st.write](#)

## st.write

8

Streamlit Version v1.9.0 ▾

Write arguments to the app.

This is the Swiss Army knife of Streamlit commands: it does different things depending on what you throw at it. Unlike other Streamlit commands, write() has some unique properties:

1. You can pass in multiple arguments, all of which will be written.
2. Its behavior depends on the input types as follows.
3. It returns None, so its "slot" in the App cannot be reused.

### Function signature

`st.write(*args, **kwargs)`

#### Parameters

One or many objects to print to the App.

Arguments are handled as follows:

- `write(string)` : Prints the formatted Markdown string, with support for LaTeX expression and emoji shortcodes. See docs for `st.markdown` for more.
- `write(data_frame)` : Displays the DataFrame as a table.
- `write(error)` : Prints an exception specially.
- `write(func)` : Displays information about a function.
- `write(module)` : Displays information about the module.
- `write(dict)` : Displays dict in an interactive widget.
- `write(mpl_fig)` : Displays a Matplotlib figure.
- `write(altair)` : Displays an Altair chart.
- `write(keras)` : Displays a Keras model.
- `write(graphviz)` : Displays a Graphviz graph.
- `write(plotly_fig)` : Displays a Plotly figure.
- `write(bokeh_fig)` : Displays a Bokeh figure.
- `write(sympy_expr)` : Prints SymPy expression using LaTeX.
- `write(htmlable)` : Prints `_repr_html_()` for the object if available.
- `write(obj)` : Prints `str(obj)` if otherwise unknown.

#### \*args (any)

`unsafe_allow_html` This is a keyword-only argument that defaults to False.  
(bool)

By default, any HTML tags found in strings will be escaped and therefore treated as pure text. This behavior may be turned off by setting this argument to True.

That said, we strongly advise against it. It is hard to write secure HTML, so by using this argument you may be compromising your users' security. For more information, see:

<https://github.com/streamlit/streamlit/issues/152>

**Also note that `unsafe\_allow\_html` is a temporary measure and may be removed from Streamlit at any time.**

If you decide to turn on HTML anyway, we ask you to please tell us your exact use case here: <https://discuss.streamlit.io/t/96>.

This will help us come up with safe APIs that allow you to do what you want.

## Example

Its basic use case is to draw Markdown-formatted text, whenever the input is a string:

```
write('Hello, *World!* :sunglasses:')
```



Hello, World! 😎

[\(view standalone Streamlit app\)](#)

As mentioned earlier, `st.write()` also accepts other data formats, such as numbers, data frames, styled data frames, and assorted objects:

```
st.write(1234)
st.write(pd.DataFrame({
    'first column': [1, 2, 3, 4],
    'second column': [10, 20, 30, 40],
}))
```



1234

	first column	second column
0	1	10
1	2	20
2	3	30
3	4	40

[\(view standalone Streamlit app\)](#)

Finally, you can pass in multiple arguments to do things like:

```
st.write('1 + 1 = ', 2)
st.write('Below is a DataFrame:', data_frame, 'Above is a dataframe.')
```



1 + 1 = 2

Below is a DataFrame:

	first column	second column
0	1	10
1	2	20
2	3	30
3	4	40

Above is a dataframe.

[\(view standalone Streamlit app\)](#)

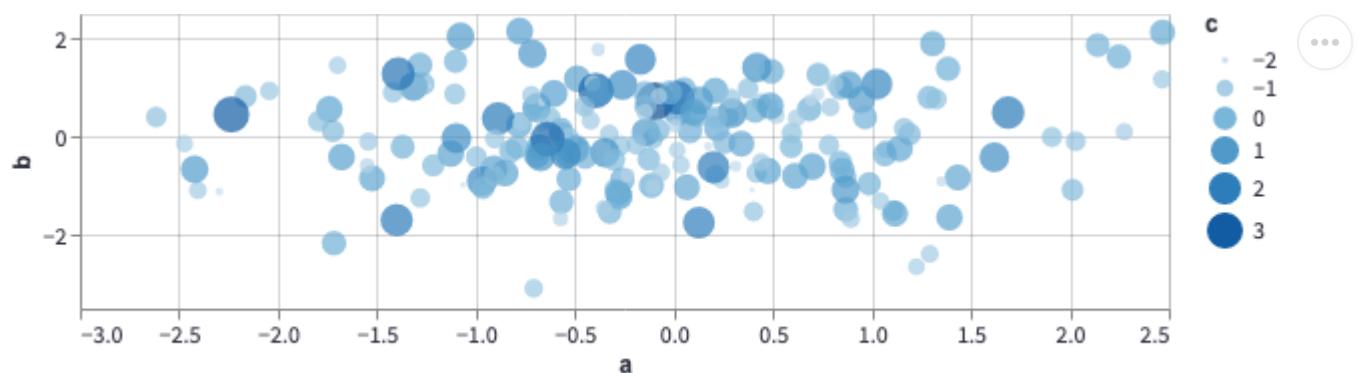
Oh, one more thing: `st.write` accepts chart objects too! For example:

```
import pandas as pd
import numpy as np
import altair as alt

df = pd.DataFrame(
    np.random.randn(200, 3),
    columns=['a', 'b', 'c'])

c = alt.Chart(df).mark_circle().encode(
    x='a', y='b', size='c', color='c', tooltip=['a', 'b', 'c'])

st.write(c)
```



[\(view standalone Streamlit app\)](#)

Was this page helpful?

Yes    No

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

←[Previous: Write and magic](#)[Next: magic](#)→

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*add*
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*add*
  - [Status elements](#)  
*remove*
    - [st.progress](#)
    - [st.spinner](#)
    - [st.balloons](#)
    - [st.snow](#)
    - [st.error](#)
    - [st.warning](#)
    - [st.info](#)
    - [st.success](#)
    - [st.exception](#)
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [\*cloud\*](#)

### [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)

- [Release notes](#)*open in new*
- [Troubleshooting](#)

- [school](#)

## [Knowledge base](#)

- [Tutorials](#)  
*add*
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home](#)/
- [Streamlit library](#)/
- [API reference](#)/
- [Status elements](#)

## **Display progress and status**

8

Streamlit provides a few methods that allow you to add animation to your apps. These animations include progress bars, status messages (like warnings), and celebratory balloons.



## **Progress bar**

Display a progress bar.

```
for i in range(101):  
    st.progress(i)  
    do_something_slow()
```



## Spinner

Temporarily displays a message while executing a block of code.

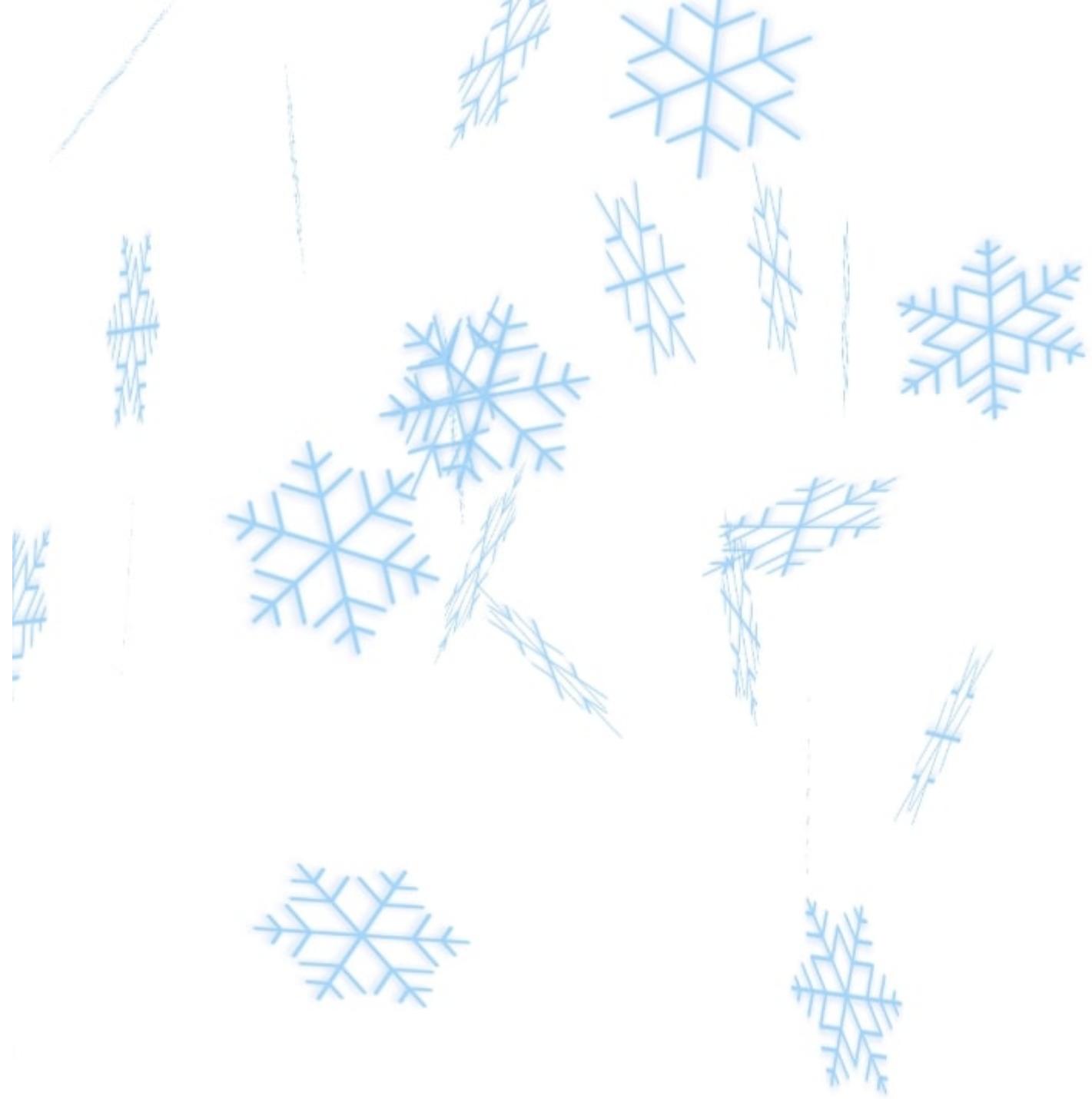
```
with st.spinner("Please wait..."):  
    do_something_slow()
```



## Balloons

Display celebratory balloons!

```
do_something()  
  
# Celebrate when all done!  
st.balloons()
```



## Snowflakes

Display celebratory snowflakes!

```
do_something()  
  
# Celebrate when all done!  
st.snow()
```

Error running analysis

## Error box

Display error message.

```
st.error("We encountered an error")
```

Trimmed input to fit buffer

## Warning box

Display warning message.

```
st.warning("Unable to fetch image. Skipping...")
```

Last datapoint is for partial data

## Info box

Display an informational message.

```
st.info("Dataset is updated every day at midnight.")
```

Analysis finished in 3.14s

## Success box

Display a success message.

```
st.success("Match found!")
```

```
ValueError: operands could not be broadcast together with shapes (10,2) (2,10)
```

Traceback:

```
File "/Users/tvst/Projects/streamlit/streamlit/docs/api-examples-source/tit  
np.random.randn(10,2) * np.random.randn(2, 10)
```

## Exception output

Display an exception.

```
e = RuntimeError("This is an exception of type RuntimeError")  
st.exception(e)
```

Was this page helpful?

*thumb\_up* Yes    *thumb\_down* No

[edit](#) [Suggest edits](#)

[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: Layouts and containers](#) [Next: st.progress](#) →

---

[Home](#) [Contact Us](#) [Community](#)





## Documentation



Search



Streamlit library

- Get started



- API reference



- Advanced features



- Components



- Changelog

- Cheat sheet



Streamlit Cloud

- Get started



- Trust and Security

- Release notes

- Troubleshooting



Knowledge base

- Tutorials



- Using Streamlit
- Streamlit Components
- Installing dependencies
- Deployment issues

Home / Streamlit Cloud

# Welcome to Streamlit Cloud

Streamlit's Community Cloud is a workspace for you and your team to easily deploy, manage, and collaborate on your Streamlit apps. If you're just getting started and have not yet built your first Streamlit app, check out the main Get started page first. When you're ready to share it, create a Community Cloud account and you can launch your app in just a few minutes! Deploy, manage, and share your apps with the world, directly from Streamlit — all for free.



## Get started

Learn how to set up your account to start deploying apps.



## Deploy an app

A step by step guide on how to get your app deployed.



## Connect data sources

Learn how to securely connect your app to data sources.



## Share your app

Share your app publicly or privately with select viewers and developers.



## Manage your app

Access logs, get more resources for your app, and other tips and tricks.

### Note

Interested in our security model? Check out our Trust and Security page.

Questions? Reach out to us on the Community forum!

Was this page helpful?

 Yes  No

 [Suggest edits](#)



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

[← Previous: Streamlit library](#)

[Next: Get started →](#)

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

[Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [\*cloud\*](#)

[Streamlit Cloud](#)

- [Get started](#)  
*remove*
  - [Deploy an app](#)  
*add*
  - [Share your app](#)
  - [Manage your app](#)
- [Trust and Security](#)
- [Release notes](#)*open in new*
- [Troubleshooting](#)

- [\*school\*](#)

[Knowledge base](#)

- [Tutorials](#)  
*add*
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home/](#)
- [Streamlit Cloud/](#)
- [Get started/](#)
- [Share your app/](#)
- [Configuring Single Sign on \(SSO\)/](#)
- [Okta](#)

## Okta Single Sign-On via SAML

8



# Sign in

[Continue with Google](#)  
[Continue with GitHub](#)  

---

OR

Your email...

[Continue](#)

By logging in, you agree to our [terms of use](#).

Okta SAML is just one of the authenticators supported by Streamlit for Teams. We have already released documentation for Microsoft ADFS, Azure AD, and generic SAML. Enabling Single Sign-On via Okta allows members of your organization to securely sign in to Streamlit using the same email address and password they already use for Okta.

## Single Sign-On via Okta for developers of your organization's apps

β

- Your developers can use Okta to log into Streamlit and access their app dashboard.
- Your developers can also give access to app viewers through their Okta logins.

## Single Sign-On via Okta for viewers of your organization's private apps

β

- Viewers added to a private app can use Okta to authenticate their identity.
- These viewers must be added to the app's viewer list by their Okta-associated email address.

## Configuring Okta SSO

β

There are three steps your team will need to complete to create an Okta SSO connection:

1. Please complete [this form](#).

To complete steps 2 and 3, you will need an **ACS URL** and **Identity Provider URI (Entity ID)**, which Streamlit will provide by emailing you a private Google Drive link. Please complete [this form](#) to provide us with your email address and some basic information about your organization.

2. Provide Streamlit with a Token Signature (X.509 Certificate).

- Follow [WorkOS' instructions to generate the token signature](#) (see "[Obtain Identity Provider Details](#)").
- Please share the Token Signature with Streamlit by uploading it [here](#).

*push\_pin*

### Note

The Token Signature is a certificate used to securely sign tokens issued by Okta.

### 3. Provide Streamlit with an Identity Provider SSO URL.

- Follow [WorkOS' instructions to generate the IdP SSO URL](#) (see "Obtain Identity Provider Details").
- Please share the IdP SSO URL with Streamlit by pasting it [here](#).

*push\_pin*

#### Note

The IdP SSO URL provides Streamlit with a login endpoint to redirect your organization's users from our login page to your Okta login page.

Was this page helpful?

Yes No

[edit](#)[Suggest edits](#)

[forum](#)

#### Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

←[Previous: Microsoft Azure ADNext: Auth0](#)→

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.



- Home/
- Knowledge base/
- Using Streamlit/
- How can I make Streamlit watch for changes in other modules I'm importing in my app?

# How can I make Streamlit watch for changes in other modules I'm importing in my app?

By default, Streamlit only watches modules contained in the current directory of the main app module. You can track other modules by adding the parent directory of each module to the `PYTHONPATH`.

```
export PYTHONPATH=$PYTHONPATH:/path/to/module  
streamlit run your_script.py
```

Was this page helpful?

Yes    No

[Suggest edits](#)

---

Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

**Next:** What browsers does Streamlit support? [→](#)

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.

## Documentation

*search*

Search

- [description](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [cloud](#)

### [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notes](#)[open in new](#)
- [Troubleshooting](#)

- [school](#)

### [Knowledge base](#)

- [Tutorials](#)  
*add*
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home](#)/
- [Knowledge base](#)/
- [Using Streamlit](#)/
- [What browsers does Streamlit support?](#)

## **What browsers does Streamlit support?**

*8*

The latest version of Streamlit is compatible with the two most recent versions of the following browsers:

- [Google Chrome](#)
- [Firefox](#)
- [Microsoft Edge](#)
- [Safari](#)

*push\_pin*

## **Note**

You may not be able to use all the latest features of Streamlit with unsupported browsers or older versions of the above browsers. Streamlit will not provide bug fixes for unsupported browsers.

Was this page helpful?

Yes  No

[edit](#)[Suggest edits](#)

[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

←[Previous: How can I make Streamlit watch for changes in other modules I'm importing in my app?](#) [Next: Where does st.file\\_uploader store uploaded files and when do they get deleted?→](#)

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [\*cloud\*](#)

### [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notes](#)*open in new*
- [Troubleshooting](#)

- [\*school\*](#)

### [Knowledge base](#)

- [Tutorials](#)  
*remove*
  - [Connect to data sources](#)  
*remove*
    - [AWS S3](#)
    - [BigQuery](#)
    - [Snowflake](#)
    - [Microsoft SQL Server](#)
    - [Firestore](#)*open in new*
    - [MongoDB](#)
    - [MySQL](#)
    - [PostgreSQL](#)
    - [Tableau](#)
    - [Private Google Sheet](#)
    - [Public Google Sheet](#)
    - [TigerGraph](#)
    - [Deta Base](#)
    - [Supabase](#)
    - [Google Cloud Storage](#)
  - [Session State basics](#)
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [\*Home\*/](#)
- [\*Knowledge base\*/](#)
- [\*Tutorials\*/](#)
- [\*Connect to data sources\*/](#)
- [\*Tableau\*](#)

## [Connect Streamlit to Tableau](#)

*8*

## **Introduction**

*8*

This guide explains how to securely access data on Tableau from Streamlit Cloud. It uses the [tableauserverclient](#) library and Streamlit's [secrets management](#).

## **Create a Tableau site**

*8*

*push\_pin*

### **Note**

If you already have a database that you want to use, feel free to [skip to the next step](#).

For simplicity, we are using the cloud version of Tableau here but this guide works equally well for self-hosted deployments. First, sign up for [Tableau Online](#) or log in. Create a workbook or run one of the example workbooks under "Dashboard Starters".

The screenshot shows the Tableau Online homepage with a sidebar on the left containing various icons for navigation and management. The main area features a search bar at the top. Below it, there's a section titled "Dashboard Starters" with a "See All" link. A message encourages users to start from pre-built workbooks. Two dashboard examples are displayed: "Salesforce" and "Marketo".

**Salesforce**

**Marketo**

## Create personal access tokens

8

While the Tableau API allows authentication via username and password, you should use [personal access tokens](#) for a production app.

Go to your [Tableau Online homepage](#), create an access token and note down the token name and secret.

The screenshot shows the Tableau Online home page. At the top right, there is a user profile icon with initials 'PS'. A red circle highlights the 'My Account Settings' option in the dropdown menu. The main content area features a large banner with the text 'Welcome to your Tableau site' and a 'Create' button. Below the banner are links for 'Upload Workbook', 'Manage Projects', 'Manage Users', 'Download Tableau Desktop', and 'Download Tableau Prep Builder'. On the left, there is a sidebar with various icons and a 'Recents' section.

The screenshot shows the 'Settings' page for a user named 'Peter Schmidt'. The 'Personal Access Tokens' section is highlighted with a red circle around the 'Create new token' button. The page also includes sections for 'Connected Clients', 'Start Page', and 'Welcome Banner'. There is an 'Interactive Tours' button at the bottom right.

*push\_pin*

**Note**

Personal access tokens will expire if not used after 15 consecutive days.

# Add token to your local app secrets

8

Your local Streamlit app will read secrets from a file `.streamlit/secrets.toml` in your app's root directory. Create this file if it doesn't exist yet and add your token, the site name you created during setup, and the URL of your Tableau server like below:

```
# .streamlit/secrets.toml

[tableau]
token_name = "xxx"
token_secret = "xxx"
server_url = "https://abc01.online.tableau.com/"
site_id = "streamliteexample" # in your site's URL behind the server_url

priority_high
```

## Important

Add this file to `.gitignore` and don't commit it to your Github repo!

## Copy your app secrets to the cloud

8

As the `secrets.toml` file above is not committed to Github, you need to pass its content to your deployed app (on Streamlit Cloud) separately. Go to the [app dashboard](#) and in the app's dropdown menu, click on **Edit Secrets**. Copy the content of `secrets.toml` into the text area. More information is available at [Secrets Management](#).

The screenshot shows the Streamlit Cloud interface. At the top, there's a navigation bar with icons for home, search, and user profile, followed by the text "awesomestreamlit's apps · Stre" and a "+" button. Below the bar, the URL "share.streamlit.io" is visible. On the left, a sidebar lists "App settings", "Sharing", and "Secrets". The "Secrets" option is selected and highlighted with a blue border. A modal window titled "Secrets" is open over the sidebar. The modal contains instructions: "Provide environment variables and other secrets to your app using TOML format. This information is encrypted and served securely to your app at runtime. Learn more about Secrets in our [docs](#). Changes take around a minute to propagate." Below the instructions is a code editor containing the following TOML code:

```
DB_USERNAME = "myuser"
DB_TOKEN = "abcdef"

[some_section]
some_key = 1234
```

In the bottom right corner of the modal, there is a blue "Save" button.

## Add `tableauserverclient` to your requirements file

8

Add the [tableauserverclient](#) package to your `requirements.txt` file, preferably pinning its version (replace `x.x.x` with the version you want installed):

```
# requirements.txt
tableauserverclient==x.x.x
```

# Write your Streamlit app

8

Copy the code below to your Streamlit app and run it. Note that this code just shows a few options of data you can get – explore the [tableauserverclient](#) library to find more!

```
# streamlit_app.py

import streamlit as st
import tableauserverclient as TSC

# Set up connection.
tableau_auth = TSC.PersonalAccessTokenAuth(
    st.secrets["tableau"]["token_name"],
    st.secrets["tableau"]["personal_access_token"],
    st.secrets["tableau"]["site_id"],
)
server = TSC.Server(st.secrets["tableau"]["server_url"], use_server_version=True)

# Get various data.
# Explore the tableauserverclient library for more options.
# Uses st.experimental_memo to only rerun when the query changes or after 10 min.
@st.experimental_memo(ttl=600)
def run_query():
    with server.auth.sign_in(tableau_auth):

        # Get all workbooks.
        workbooks, pagination_item = server.workbooks.get()
        workbooks_names = [w.name for w in workbooks]

        # Get views for first workbook.
        server.workbooks.populate_views(workbooks[0])
        views_names = [v.name for v in workbooks[0].views]

        # Get image & CSV for first view of first workbook.
        view_item = workbooks[0].views[0]
        server.views.populate_image(view_item)
        server.views.populate_csv(view_item)
        view_name = view_item.name
        view_image = view_item.image
        # `view_item.csv` is a list of binary objects, convert to str.
        view_csv = b"".join(view_item.csv).decode("utf-8")

        return workbooks_names, views_names, view_name, view_image, view_csv

workbooks_names, views_names, view_name, view_image, view_csv = run_query()

# Print results.
st.subheader("Workbook")
st.write("Found the following workbooks:", ", ".join(workbooks_names))

st.subheader("View")
st.write(
    f"Workbook *{workbooks_names[0]}* has the following views:",
    ", ".join(views_names),
)

st.subheader("Image")
st.write(f"Here's what view *{view_name}* looks like:")
st.image(view_image, width=300)

st.subheader("Data")
st.write(f"And here's the data for view *{view_name}*:")
st.write(pd.read_csv(StringIO(view_csv)))
```

See `st.experimental_memo` above? Without it, Streamlit would run the query every time the app reruns (e.g. on a widget interaction). With `st.experimental_memo`, it only runs when the query changes or after 10 minutes (that's what `ttl` is for). Watch out: If your database updates more frequently, you should adapt `ttl` or remove caching so viewers always see the latest data. Read more about caching [here](#).

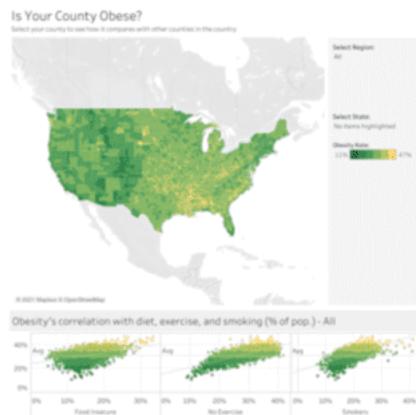
If everything worked out, your app should look like this (can differ based on your workbooks):

## Views

Workbook *Regional* has the following views: Obesity, College, Global Temperatures, Flight Delays, Economy, Stocks

## Image

Here's what view *Obesity* looks like:



## Data

And here's the data for view *Obesity*:

	County	Region	State	Above or Below?	Avg. Adult Obesi
0	Dane	Midwest	Wisconsin	10% below	
1	Johnson	Midwest	Iowa	9% below	
2	Hennepin	Midwest	Minnesota	9% below	
3	Jackson	Midwest	Kansas	0% below	

Was this page helpful?

Yes  No

[edit](#)[Suggest edits](#)

[forum](#)

**Still have questions?**

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: PostgreSQL](#) [Next: Private Google Sheet](#) →

[Home](#) [Contact Us](#) [Community](#)





# Text elements

Streamlit apps usually start with a call to `st.title` to set the app's title. After that, there are 2 heading levels you can use: `st.header` and `st.subheader`.

Pure text is entered with `st.text`, and Markdown with `st.markdown`.

We also offer a "swiss-army knife" command called `st.write`, which accepts multiple arguments, and multiple data types. And as described above, you can also use magic commands in place of `st.write`.

## Tincidunt lobortis

Feugiat vivamus at augue eget arcu dictum. Sed risus pretium quam vulputate. Cursus in hac habitasse platea dictumst. Aliquam ultrices sagittis orci a.

## Non diam phasellus vestibulum

Vel quam elementum pulvinar etiam. Blandit volutpat maecenas volutpat blandit aliquam. Est sit amet

Display string

formatted as

Markdown.

`st.markd`

**THIS IS A TITLE**  
**This is a title**  
**This is a title**

Display text in  
title formatting.

Title

`st.title(`

**This is a header**

This is a header

# This is a header

## This is a header

Display text in  
header  
formatting.

st.header

Header

### This is a subheader

### This is a subheader

### This is a subheader

Display text in  
subheader  
formatting.

st.subhea

Subheader

This is a caption

This is a caption

This is a caption

Display text in  
small font.

st.captic

Caption

```
if current_frame.f_back is not None:  
    lines = inspect.getframeinfo(current_frame.f_back)[3]  
else:
```

```
    lines = None

if not lines:
    warning("`show` not enabled in the shell")
    return
```

Display a code block with optional syntax highlighting.

Write fixed-width and preformatted text.

$$E' = \nabla \times B - 4\pi j$$

$$G_{\mu\nu} + \Lambda g_{\mu\nu} = \frac{8\pi G}{c^4} T_{\mu\nu}$$

Display mathematical expressions formatted as st. `latex()` `\LaTeX{}`.

 [Suggest edits](#)

---

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[← Previous: Write and magic](#)

[Next: st.markdown →](#)

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*add*
- [Advanced features](#)  
*remove*
  - [Theming](#)
  - [Configuration](#)
  - [Optimize performance with st.cache](#)
  - [Experimental cache primitives](#)
  - [Add statefulness to apps](#)
  - [Widget semantics](#)
  - [Pre-release features](#)
  - [Working with timezones](#)
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [\*cloud\*](#)

### [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notes](#)*open in new*
- [Troubleshooting](#)

- [\*school\*](#)

### [Knowledge base](#)

- [Tutorials](#)  
*add*
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home/](#)
- [Streamlit library/](#)
- [Advanced features/](#)
- [Theming](#)

## [Theming](#)

In this guide, we provide examples of how Streamlit page elements are affected by the various theme config options. For a more high-level overview of Streamlit themes, see the Themes section of the [main concepts documentation](#).

Streamlit themes are defined using regular config options: a theme can be set via command line flag when starting your app using `streamlit run` or by defining it in the `[theme]` section of a `.streamlit/config.toml` file. For more information on setting config options, please refer to the [Streamlit configuration documentation](#).

The following config options show the default Streamlit Light theme recreated in the `[theme]` section of a `.streamlit/config.toml` file.

```
[theme]
primaryColor="#F63366"
backgroundColor="#FFFFFF"
secondaryBackgroundColor="#F0F2F6"
textColor="#262730"
font="sans serif"
```

Let's go through each of these options, providing screenshots to demonstrate what parts of a Streamlit app they affect where needed.

## [primaryColor](#)

`primaryColor` defines the accent color most often used throughout a Streamlit app. A few examples of Streamlit widgets that use `primaryColor` include `st.checkbox`, `st.slider`, and `st.text_input` (when focused).

Check me

A slider

83

0 100

focused text input

foo

star

Tip

Any CSS color can be used as the value for primaryColor and the other color options below. This means that theme colors can be specified in hex or with browser-supported color names like "green", "yellow", and "chartreuse". They can even be defined in the RGB and HSL formats!

## backgroundColor

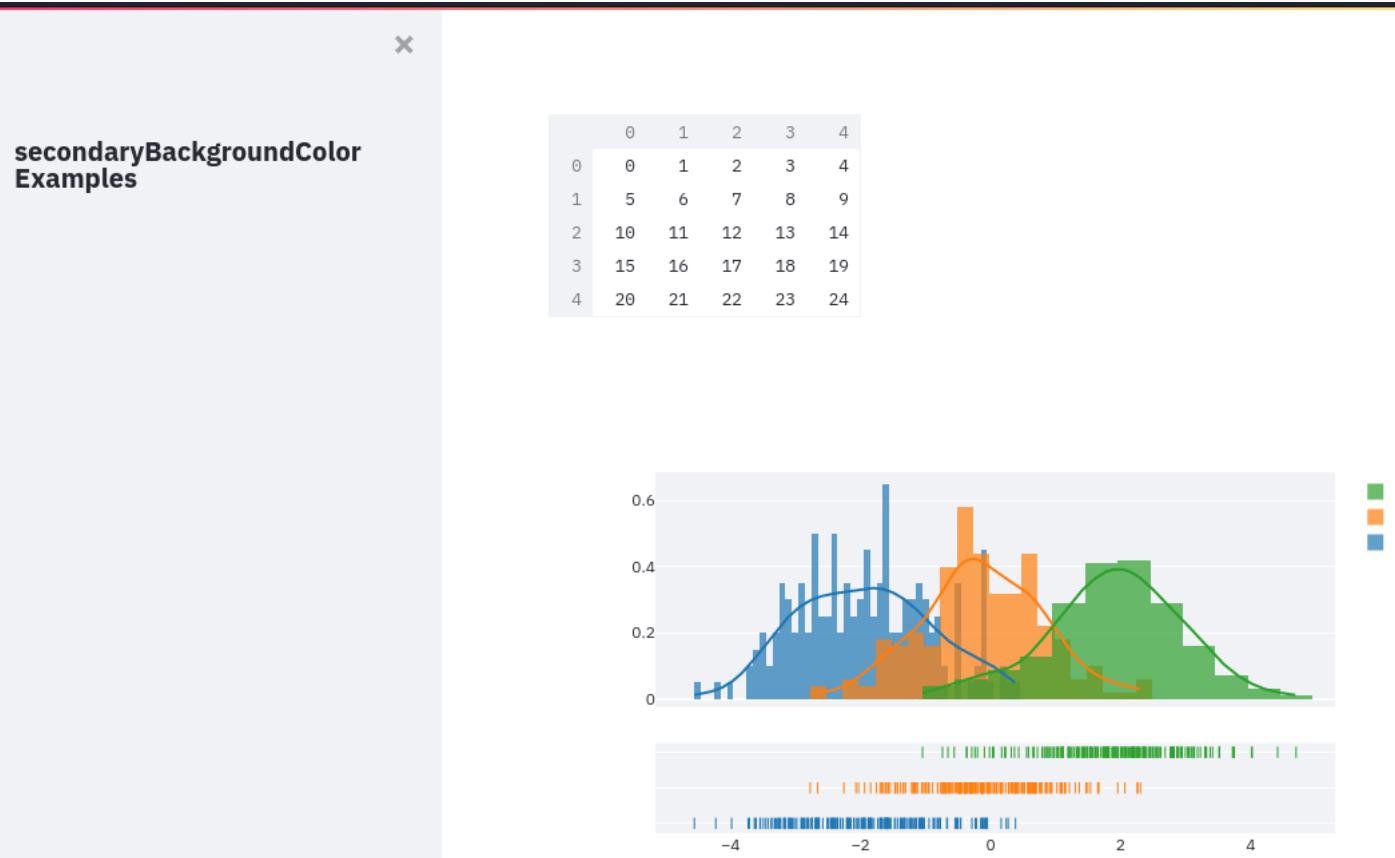
8

Defines the background color used in the main content area of your app.

## secondaryBackgroundColor

8

This color is used where a second background color is needed for added contrast. Most notably, it is the sidebar's background color. It is also used as the plot background color for `st.plotly_chart` and as the background color for most other interactive widgets.



## textColor

8

This option controls the text color for most of your Streamlit app.

## font

8

Selects the font used in your Streamlit app. Valid values are `"sans serif"`, `"serif"`, and `"monospace"`. This option defaults to `"sans serif"` if unset or invalid.

Note that code blocks are always rendered using the monospace font regardless of the font selected here.

## base

8

An easy way to define custom themes that make small changes to one of the preset Streamlit themes is to use the `base` option. Using `base`, the Streamlit Light theme can be recreated as a custom theme by writing the following:

```
[theme]  
base="light"
```

The `base` option allows you to specify a preset Streamlit theme that your custom theme inherits from. Any theme config options not defined in your theme settings have their values set to those of the base theme. Valid values for `base` are "light" and "dark".

For example, the following theme config defines a custom theme nearly identical to the Streamlit Dark theme, but with a new `primaryColor`.

```
[theme]  
base="dark"  
primaryColor="purple"
```

If `base` itself is omitted, it defaults to "light", so you can define a custom theme that changes the font of the Streamlit Light theme to serif with the following config

```
[theme]  
font="serif"
```

Was this page helpful?

Yes No  
[edit](#)[Suggest edits](#)  
[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

[←Previous: Advanced features](#)[Next: Configuration→](#)

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.



# Connect Streamlit to TigerGraph

8

## Introduction

8

This guide explains how to securely access a TigerGraph database from Streamlit Cloud. It uses the [pyTigerGraph](#) library and Streamlit's [secrets management](#).

## Create a TigerGraph Cloud Database

8

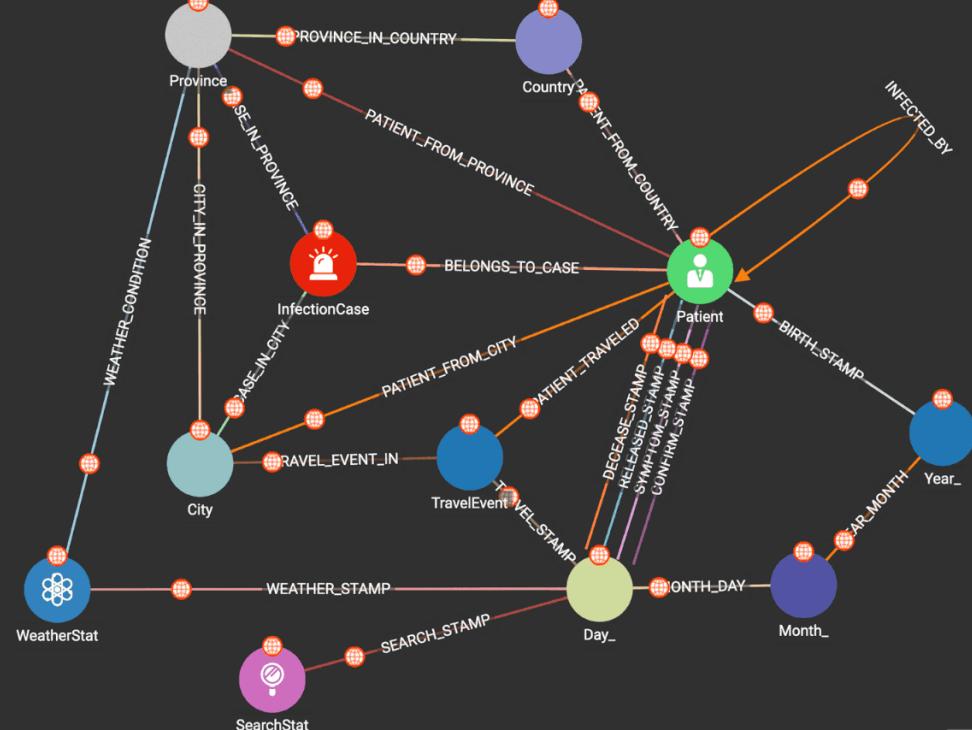
First, follow the official tutorials to create a TigerGraph instance in TigerGraph Cloud, either as a [blog](#) or a [video](#). Note your username, password, and subdomain.

For this tutorial, we will be using the COVID-19 starter kit. When setting up your solution, select the "COVID-19 Analysis" option.

The screenshot shows the TigerGraph Cloud interface for creating a new solution. The left sidebar includes links for Dashboard, My Solutions, My Activities, Network, and My Account. The main area has a header 'Welcome to TigerGraph Cloud!' and a 'Create Solution' button. A progress bar indicates four steps: 1 Basic Settings, 2 Instance Settings, 3 Solution Settings, and 4 Confirmation. Step 1 is active. A dropdown for 'Select TigerGraph Version' is set to '3.1.5'. Below this, a section titled 'Select a Starter Kit' lists various options. The 'COVID-19 Analysis v3.1.5' kit is highlighted with an orange border. Other kits shown include 'Blank v3.1.5', 'Customer 360 - Attribution and Engagement Graph v3.1.5', 'Cybersecurity Threat Detection - IT v3.1.5', 'Data Lineage v3.1.5', 'Enterprise Knowledge Graph (Corporate) v3.1.5', 'Enterprise Knowledge Graph (Crunchbase) v3.1.5', 'Entity Resolution (MDM) v3.1.5', 'Financial Services (Payments) - Fraud Detection v3.1.5', 'Fraud and Money Laundering Detection (Fin. Services) v3.1.5', 'GSQL 101 v3.1.5', and 'Graph Analytics - Centrality Algorithms v3.1.5'. Each kit has a small icon and an information button.

Once it is started, ensure your data is downloaded and queries are installed.

- M MyGraph  
superuser
- Design Schema
- X Map Data To Graph
- Load Data
- Explore Graph
- B Build Graph Patterns BETA
- W Write Queries



q

## Add username and password to your local app secrets

8

Your local Streamlit app will read secrets from a file `.streamlit/secrets.toml` in your app's root directory. Create this file if it doesn't exist yet and add your TigerGraph Cloud instance username, password, graph name, and subdomain as shown below:

```
# .streamlit/secrets.toml
```

```
[  
tigergraph  
]  
host = "https://xxx.i.tgcloud.io/"  
username = "xxx"  
password = "xxx"  
graphname = "xxx"
```

Copy

!

### Important

Add this file to `.gitignore` and don't commit it to your Github repo!

## Copy your app secrets to the cloud

As the `secrets.toml` file above is not committed to Github, you need to pass its content to your deployed app (on Streamlit Cloud) separately. Go to the [app dashboard](#) and in the app's dropdown menu, click on Edit Secrets. Copy the content of `secrets.toml` into the text area. More information is available at [Secrets Management](#).

The screenshot shows a web browser window with the URL `share.streamlit.io`. The page title is "awesomestreamlit's apps". On the left, there is a sidebar with three options: "App settings", "Sharing", and "Secrets", where "Secrets" is currently selected. A modal window titled "Secrets" is open in the center. It contains instructions about providing environment variables and secrets in TOML format, mentioning encryption and runtime delivery. Below the instructions is a text area containing the following TOML code:

```
DB_USERNAME = "myuser"
DB_TOKEN = "abcdef"

[some_section]
some_key = 1234
```

At the bottom right of the modal is a blue "Save" button.

## Add PyTigerGraph to your requirements file

8

Add the pyTigerGraph package to your `requirements.txt` file, preferably pinning its version (replace `x.x.x` with the version you want installed):

```
# requirements.txt
pyTigerGraph==x.x.x
```

Copy

## Write your Streamlit app

Copy the code below to your Streamlit app and run it. Make sure to adapt the name of your graph and query.

```
# streamlit_app.py

import streamlit as st
import pyTigerGraph as tg

# Initialize connection.
conn = tg.TigerGraphConnection(**st.secrets["tigergraph"])
conn.apiToken = conn.getToken(conn.createSecret())

# Pull data from the graph by running the "mostDirectInfections" query.
# Uses st.experimental_memo to only rerun when the query changes or after 10 min.
@st.experimental_memo(ttl=600)
def get_data():
    most_infections = conn.runInstalledQuery("mostDirectInfections") [0] ["Answer"] [0]
    return most_infections["v_id"], most_infections["attributes"]

items = get_data()

# Print results.
st.title(f"Patient {items[0]} has the most direct infections")
for key, val in items[1].items():
    st.write(f"Patient {items[0]}'s {key} is {val}.")
```

Copy

See `st.experimental_memo` above? Without it, Streamlit would run the query every time the app reruns (e.g. on a widget interaction). With `st.experimental_memo`, it only runs when the query changes or after 10 minutes (that's what `ttl` is for). Watch out: If your database updates more frequently, you should adapt `ttl` or remove caching so viewers always see the latest data. Read more about caching [here](#).

If everything worked out (and you used the example data we created above), your app should look like this:

# Patient 2000000205 has the most direct infections

Patient 2000000205's patient\_id is 2000000205.

Patient 2000000205's global\_num is 8100.

Patient 2000000205's birth\_year is 1946.

Patient 2000000205's infection\_case is contact with patient.

Patient 2000000205's contact\_number is 8.

Patient 2000000205's symptom\_onset\_date is 1970-01-01 00:00:00.

Patient 2000000205's confirmed\_date is 2020-03-14 00:00:00.

Patient 2000000205's released\_date is 1970-01-01 00:00:00.

Patient 2000000205's deceased\_date is 1970-01-01 00:00:00.

Patient 2000000205's state is isolated.

Patient 2000000205's disease is .

Patient 2000000205's sex is female.

Patient 2000000205's @directInfections is 51.

Made with Streamlit



Was this page helpful?

Yes No

Suggest edits



## Still have questions?

Our forums are full of helpful information and Streamlit experts.



Previous: Public Google Sheet



Next: Data Base



[Home](#)   [Contact Us](#)   [Community](#)



Copyright © 2022, Streamlit Inc.

## Documentation

*search*

Search

- *description*

### Streamlit library

- Get started  
*add*
- API reference  
*add*
- Advanced features  
*add*
- Components  
*add*
- Changelog
- Cheat sheet

- *cloud*

### Streamlit Cloud

- Get started  
*add*
- Trust and Security
- Release notes*open in new*
- Troubleshooting

- *school*

### Knowledge base

- Tutorials  
*remove*
  - Connect to data sources  
*add*
  - Session State basics
- Using Streamlit
- Streamlit Components
- Installing dependencies
- Deployment issues

- *Home/*
- *Knowledge base/*
- *Tutorials*

## **Tutorials**

Our tutorials include step-by-step examples of building different types of apps in Streamlit.

- [Connect to data sources](#)
- [Session State basics](#)

Was this page helpful?

Yes  No

[edit](#) [Suggest edits](#)

[forum](#)

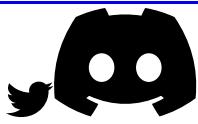
## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: Knowledge base](#) [Next: Connect to data sources](#) →

---

[Home](#) [Contact Us](#) [Community](#)



Copyright © 2022, Streamlit Inc.

## Documentation

*search*

Search

- [description](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [cloud](#)

### [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notes](#)[open in new](#)
- [Troubleshooting](#)

- [school](#)

### [Knowledge base](#)

- [Tutorials](#)  
*add*
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home](#)/

- [Knowledge base](#)/

- [Using Streamlit](#)

## **Using Streamlit**

8

Here are some frequently asked questions about using Streamlit. If you feel something important is missing that everyone needs to know, please [open an issue](#) or [submit a pull request](#) and we'll be happy to review it!

- [Sanity checks](#)
- [Caching issues](#)
- [Batch elements and input widgets with `st.form`](#)
- [How do I run my Streamlit script?](#)
- [How can I make Streamlit watch for changes in other modules I'm importing in my app?](#)
- [What browsers does Streamlit support?](#)
- [What is the path of Streamlit's `config.toml` file?](#)
- [Where does `st.file\_uploader` store uploaded files and when do they get deleted?](#)
- [How do you retrieve the filename of a file uploaded with `st.file\_uploader`?](#)
- [How to remove "Streamlit" from the app title?](#)

- [How to download a file in Streamlit?](#)
- [How to download a Pandas DataFrame as a CSV?](#)
- [How can I make `st.pydeck\_chart` use custom Mapbox styles?](#)
- [How to insert elements out of order?](#)
- [How to animate elements?](#)
- [Append data to a table or chart](#)
- [Hide row indices when displaying a dataframe](#)
- [How to record a screencast?](#)
- [How do I upgrade to the latest version of Streamlit?](#)
- [How do I hide the hamburger menu from my app?](#)
- [Widget updating for every second input when using session state](#)
- [How do I create an anchor link?](#)
- [How do I enable camera access?](#)

Was this page helpful?

Yes    No

[edit](#)[Suggest edits](#)

[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

←[Previous: Tutorials](#)[Next: How to animate elements?](#)→

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.

## Documentation

*search*

Search

- *description*

### Streamlit library

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*add*
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*add*
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*remove*
    - [st.set\\_page\\_config](#)
    - [st.echo](#)
    - [st.help](#)
    - [st.experimental\\_show](#)
    - [st.experimental\\_get\\_query\\_params](#)
    - [st.experimental\\_set\\_query\\_params](#)
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- *cloud*

### Streamlit Cloud

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notes](#)*open in new*
- [Troubleshooting](#)

- *school*

## Knowledge base

- o [Tutorials](#)  
*add*
- o [Using Streamlit](#)
- o [Streamlit Components](#)
- o [Installing dependencies](#)
- o [Deployment issues](#)

- [Home/](#)
- [Streamlit library/](#)
- [API reference/](#)
- [Utilities](#)

## Placeholders, help, and options

8

There are a handful of methods that allow you to create placeholders in your app, provide help using doc strings, get and modify configuration options and query parameters.

### Set page title, favicon, and more

Configures the default settings of the page.

```
st.set_page_config(  
    title="My app",  
    favicon=":shark:",  
)
```

### Echo

Display some code on the app, then execute it. Useful for tutorials.

```
with st.echo():  
    st.write('This code will be printed')
```

### Get help

Display object's doc string, nicely formatted.

```
st.help(st.write)  
st.help(pd.DataFrame)
```

### st.experimental\_show

Write arguments and argument names to your app for debugging purposes.

```
df = pd.DataFrame({  
    'first column': [1, 2, 3, 4],  
    'second column': [10, 20, 30, 40],  
})  
st.experimental_show(df)
```

### Get query paramters

Return the query parameters that are currently showing in the browser's URL bar.

```
st.experimental_get_query_params()
```

### Set query paramters

Set the query parameters that are shown in the browser's URL bar.

```
st.experimental_set_query_params(  
    show_map=True,  
    selected=["asia"]  
)
```

Was this page helpful?

Yes  No

[edit](#)[Suggest edits](#)

[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: Control flow](#) [Next: st.set\\_page\\_config](#) →

---

[Home](#)[Contact Us](#)[Community](#)



Copyright © 2022, Streamlit Inc.



Where does `st.file_uploader` store uploaded files and when do they get deleted?

# Where does `st.file_uploader` store uploaded files and when do they get deleted?

When you upload a file using `st.file_uploader`, the data are copied to the Streamlit backend via the browser, and contained in a BytesIO buffer in Python memory (i.e. RAM, not disk). The data will persist in RAM until the Streamlit app re-runs from top-to-bottom, which is on each widget interaction. If you need to save the data that was uploaded between runs, then you can `cache` it so that Streamlit persists it across re-runs.

As files are stored in memory, they get deleted immediately as soon as they're not needed anymore.

This means Streamlit removes a file from memory when:

- The user uploads another file, replacing the original one
- The user clears the file uploader
- The user closes the browser tab where they uploaded the file

Related forum posts:

- <https://discuss.streamlit.io/t/streamlit-sharing-fileupload-where-does-it-go/9267>
- <https://discuss.streamlit.io/t/how-to-update-the-uploaded-file-using-file-uploader/13512/>

Was this page helpful?

👍 Yes

👎 No

Suggest edits



## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

---

[← Previous:](#) What browsers does Streamlit support?

**Next:** Widget updating for every second input when using session state →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.



# Advanced notes on widget behavior

Widgets are magical and often work how you want. But they can have surprising behavior in some situations. Here is a high-level, abstract description of widget behavior, including some common edge-cases:

1. If you call a widget function before the widget state exists, the widget state defaults to a value. This value depends on the widget and its arguments.
2. A widget function call returns the current widget state value. The return value is a simple Python type, and the exact type depends on the widget and its arguments.
3. Widget states depend on a particular session (browser connection). The actions of one user do not affect the widgets of any other user.
4. A widget's identity depends on the arguments passed to the widget function. If those change, the call will create a new widget (with a default value, per 1).
5. If you don't call a widget function in a script run, we neither store the widget state nor render the widget. If you call a widget function with the same arguments later, Streamlit treats it as a new widget.

4 and 5 are the most likely to be surprising and may pose a problem for some application designs. When you want to persist widget state for recreating a widget, use [Session State](#) to work around 5.

## Was this page helpful?

Yes

No

[Suggest edits](#)

Still have questions?



Our [forums](#) are full of helpful information and Streamlit experts.

---

← **Previous:** Add statefulness to apps

**Next:** Pre-release features →

---

[Home](#)    [Contact Us](#)    [Community](#)



Copyright © 2022, Streamlit Inc.

## [Documentation](#)

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*add*
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*remove*
    - [st.button](#)
    - [st.download\\_button](#)
    - [st.checkbox](#)
    - [st.radio](#)
    - [st.selectbox](#)
    - [st.multiselect](#)
    - [st.slider](#)
    - [st.select\\_slider](#)
    - [st.text\\_input](#)
    - [st.number\\_input](#)
    - [st.text\\_area](#)
    - [st.date\\_input](#)
    - [st.time\\_input](#)
    - [st.file\\_uploader](#)
    - [st.camera\\_input](#)
    - [st.color\\_picker](#)
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*add*
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)

- [Cheat sheet](#)

- [\*cloud\*](#)

## [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notes](#)*open in new*
- [Troubleshooting](#)

- [\*school\*](#)

## [Knowledge base](#)

- [Tutorials](#)  
*add*
- [Using Streamlit](#)
- [Streamlit Components](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home](#)/
- [Streamlit library](#)/
- [API reference](#)/
- [Input widgets](#)

## **Input widgets**

8

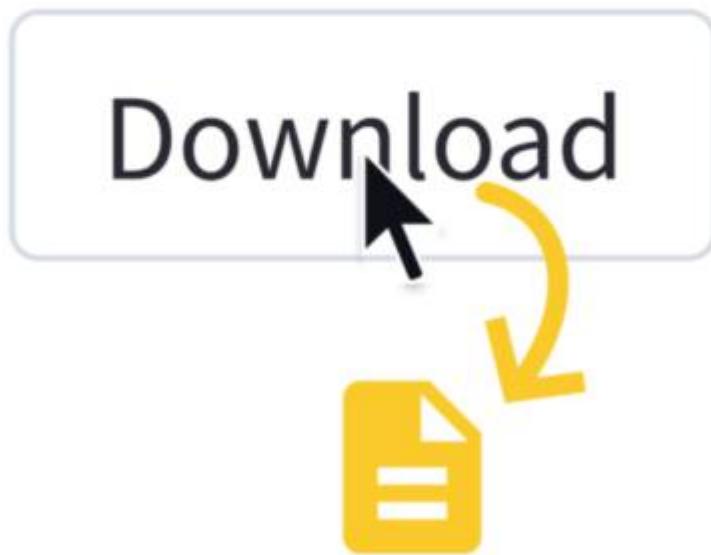
With widgets, Streamlit allows you to bake interactivity directly into your apps with buttons, sliders, text inputs, and more.



## Button

Display a button widget.

```
clicked = st.button("Click me")
```



## Download button

Display a download button widget.

```
st.download_button("Download file", file)
```



Rebuild model each time



## Checkbox

Display a checkbox widget.

```
selected = st.checkbox("I agree")
```

## Classify image

- Dog
- Cat
- Goldfish

## Radio

Display a radio button widget.

```
choice = st.radio("Pick one", ["cats", "dogs"])
```

Pick one

```
cats
```

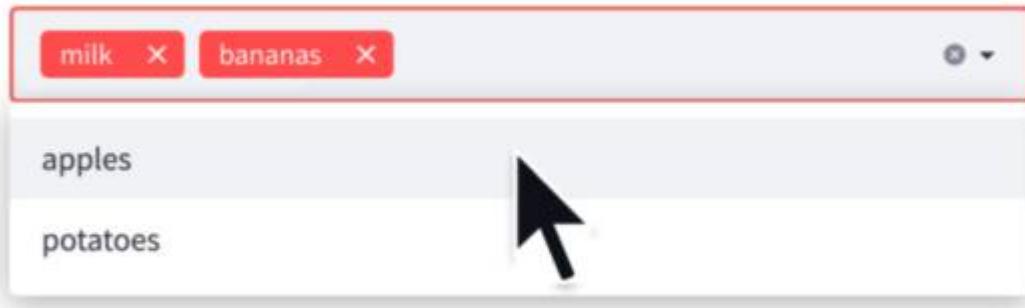
- cats
- dogs

## Selectbox

Display a select widget.

```
choice = st.selectbox("Pick one", ["cats", "dogs"])
```

Visible in image

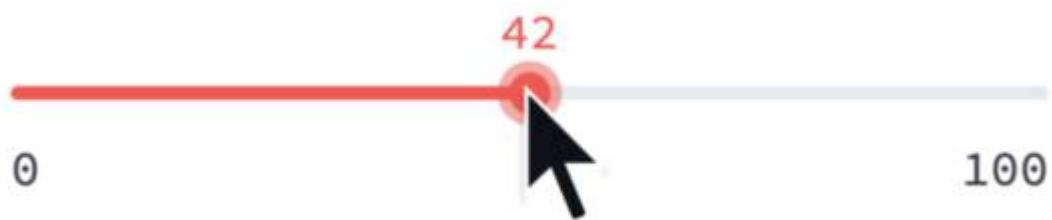


## Multiselect

Display a multiselect widget. The multiselect widget starts as empty.

```
choices = st.multiselect("Buy", ["milk", "apples", "potatoes"])
```

Pick a number

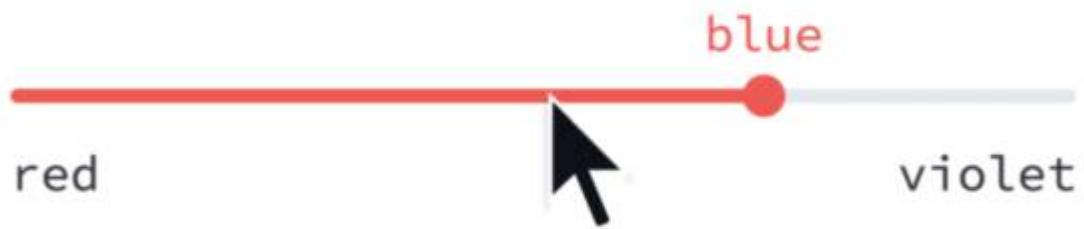


## Slider

Display a slider widget.

```
number = st.slider("Pick a number", 0, 100)
```

Dominant color in image



## Select-slider

Display a slider widget to select items from a list.

```
size = st.select_slider("Pick a size", ["S", "M", "L"])
```

Movie title

Life of Brian



## Text input

Display a single-line text input widget.

```
name = st.text_input("First name")
```

Number of days

28

- +

## Number input

Display a numeric input widget.

```
choice = st.number_input("Pick a number", 0, 10)
```

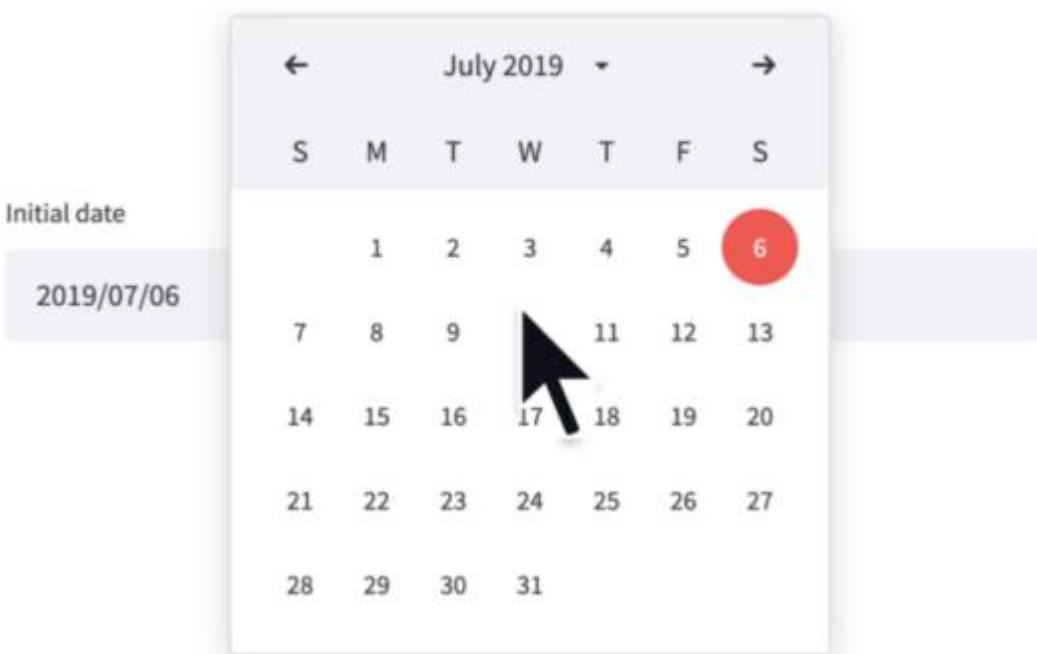
Text to analyze

It was the best of times, it was the worst of times, it was the age of wisdom,  
it was the age of foolishness, it was the epoch of belief, it was the epoch of  
incredulity, it was the season of Light  as the season of Darkness, it was  
the spring of hope, it was the winter of despair / .

## Text-area

Display a multi-line text input widget.

```
text = st.text_area("Text to translate")
```



## Date input

Display a date input widget.

```
date = st.date_input("Your birthday")
```

Set an alarm for

08:45

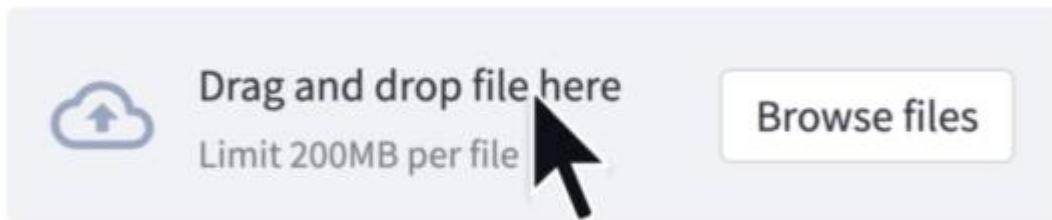


## Time input

Display a time input widget.

```
time = st.time_input("Meeting time")
```

Choose a CSV file

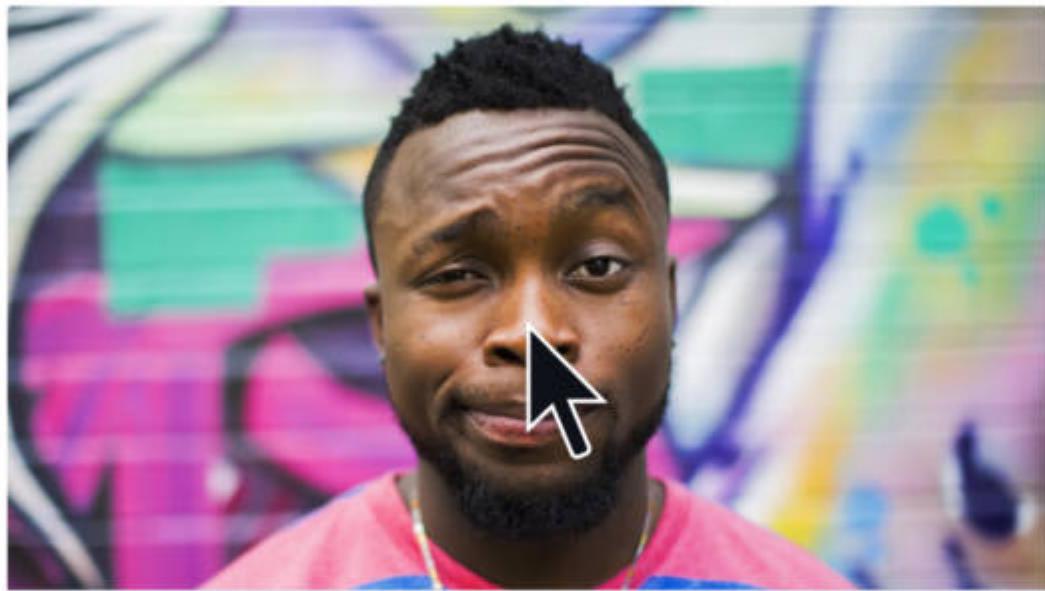


## File Uploader

Display a file uploader widget.

```
data = st.file_uploader("Upload a CSV")
```

Take a picture of an object to recognize

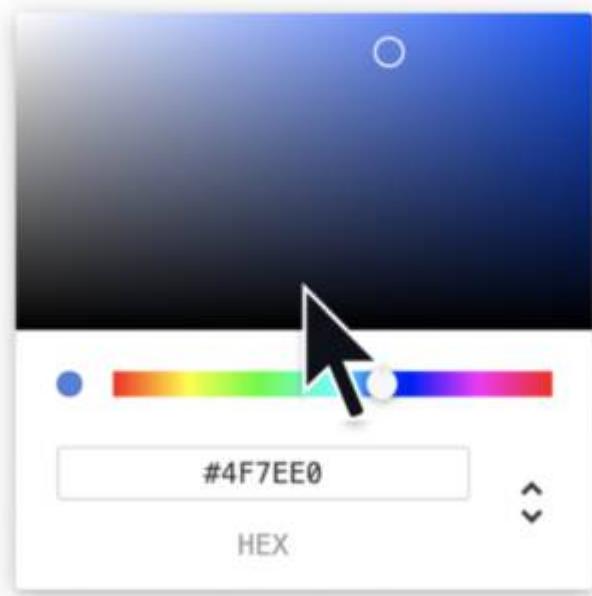


Take photo

## Camera input

Display a widget that allows users to upload images directly from a camera.

```
image = st.camera_input("Take a picture")
```



## Color picker

Display a color picker widget.

```
color = st.color_picker("Pick a color")
```

Was this page helpful?

Yes  No  
[edit](#) [Suggest edits](#)  
[forum](#)

## Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: Chart elements](#) [Next: st.button](#) →

---

[Home](#) [Contact Us](#) [Community](#)



## Documentation

*search*

Search

- [\*description\*](#)

### [Streamlit library](#)

- [Get started](#)  
*add*
- [API reference](#)  
*remove*
  - [Write and magic](#)  
*remove*
    - [st.write](#)
    - [magic](#)
  - [Text elements](#)  
*add*
  - [Data display elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*add*
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*add*
  - [Status elements](#)  
*add*
  - [Control flow](#)  
*add*
  - [Utilities](#)  
*add*
  - [Mutate charts](#)
  - [State management](#)
  - [Performance](#)  
*add*
- [Advanced features](#)  
*add*
- [Components](#)  
*add*
- [Changelog](#)
- [Cheat sheet](#)

- [\*cloud\*](#)

### [Streamlit Cloud](#)

- [Get started](#)  
*add*
- [Trust and Security](#)
- [Release notes](#)[open in new](#)
- [Troubleshooting](#)

- [\*school\*](#)

### [Knowledge base](#)

- [Tutorials](#)

- add
  - [Using Streamlit](#)
  - [Streamlit Components](#)
  - [Installing dependencies](#)
  - [Deployment issues](#)

- [Home/](#)
- [Streamlit library/](#)
- [API reference/](#)
- [Write and magic](#)

## st.write and magic commands

8

Streamlit has two easy ways to display information into your app, which should typically be the first thing you try: `st.write` and `magic`.

### st.write

Write arguments to the app.

```
st.write("Hello **world**!")  
st.write(my_data_frame)  
st.write(my_mpl_figure)
```

### Magic

Any time Streamlit sees either a variable or literal value on its own line, it automatically writes that to your app using `st.write`

```
"Hello **world**!"  
my_data_frame  
my_mpl_figure
```

Was this page helpful?

Yes  No  
[edit](#) [Suggest edits](#)  
[forum](#)

### Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

← [Previous: API reference](#) [Next: st.write](#) →

---

[Home](#) [Contact Us](#) [Community](#)



Copyright © 2022, Streamlit Inc.