Searchable Django Applications with Elastic App Search

Seth Michael Larson

@sethmlarson

All about the Workshop Materials

Examples can be followed either live or later

- On video I'll show mostly in App Search UI
- Code examples mostly replicate what's done in UI

Example are numbered, will mention to synchronize

- Example #1 corresponds to "`example_1_making_requests.py`"

All code examples available on GitHub

- `elastic/pycon-2021-workshop-app-search`

elastic

What is Elastic App Search?

elastic

Where are we in the Elastic Stack?

Product

App Search

Solution

Enterprise Search

Stack

Kibana

Elasticsearch

elastic

What is Elastic App Search?

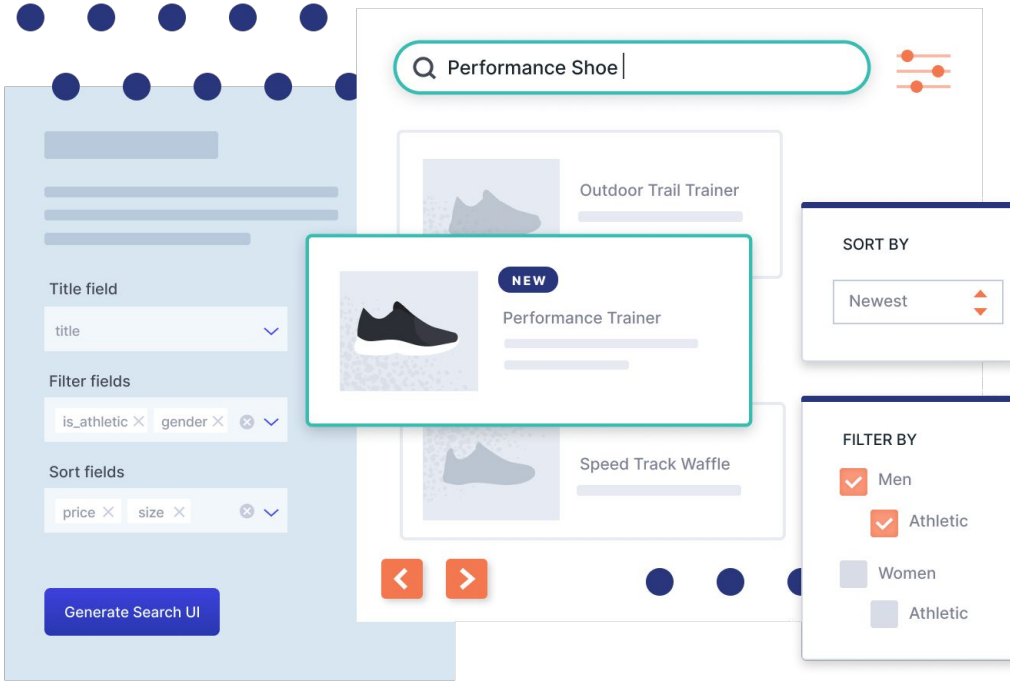
“Advanced Search made Simple”

Refined indexing and search APIs

- Schema or schema-less
- Pre-optimized search relevance

Batteries included

- Intuitive pre-built dashboards
- Web crawler
- Relevance tuning
- Analytics and insights



elastic

Conceptualizing Elastic App Search

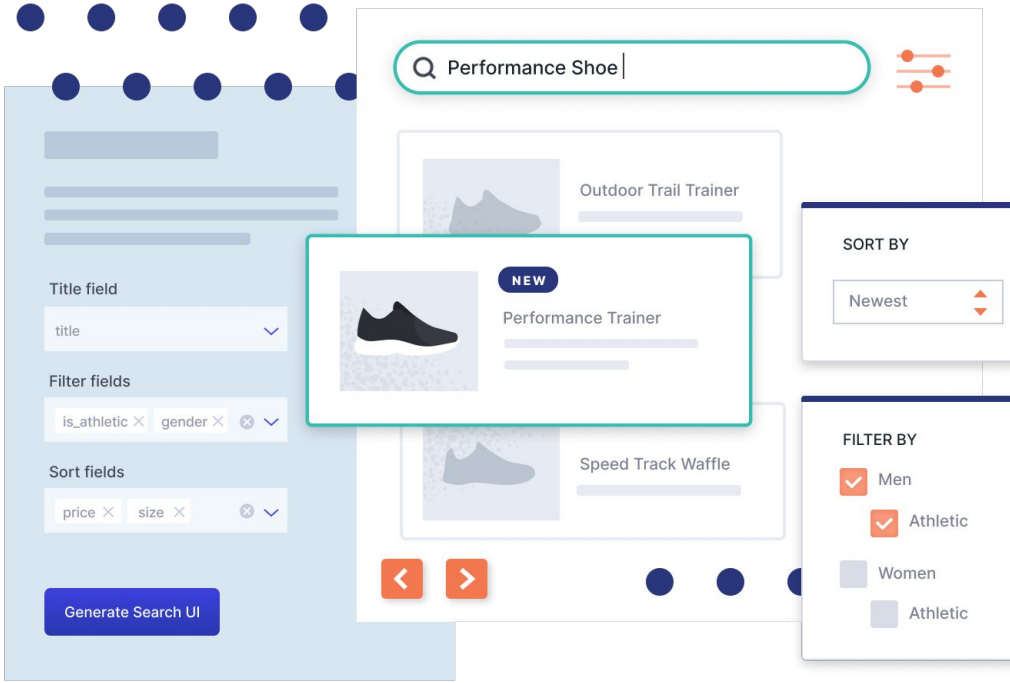
Engines and Documents

Documents in a searchable Engine

- "An engine per search bar"
- Each search result is a document

Schema, Search Settings, Analytics

Data is stored and indexed in Elasticsearch



elastic

What data is App Search for?

Create Update Delete

Search relevance is always up-to-date with documents

- Engine schema, curations, and tuning can be updated any time with zero downtime.
- Relevance tuning integrates seamlessly into search

Denormalized

Each document maps to one search result

- Each document should stand alone
- Can be a primary data-store if ACID not required

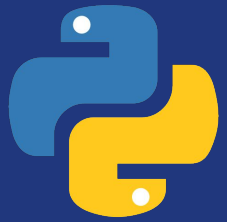
Timeless

Elasticsearch better suited for data like logs and events

- Data in the engine is available for search
- Elasticsearch better equipped for time-series data

elastic

Connecting to App Search with Python



elastic

Python Client for App Search

Package available on PyPI

```
$ python -m pip install elastic-enterprise-search
```

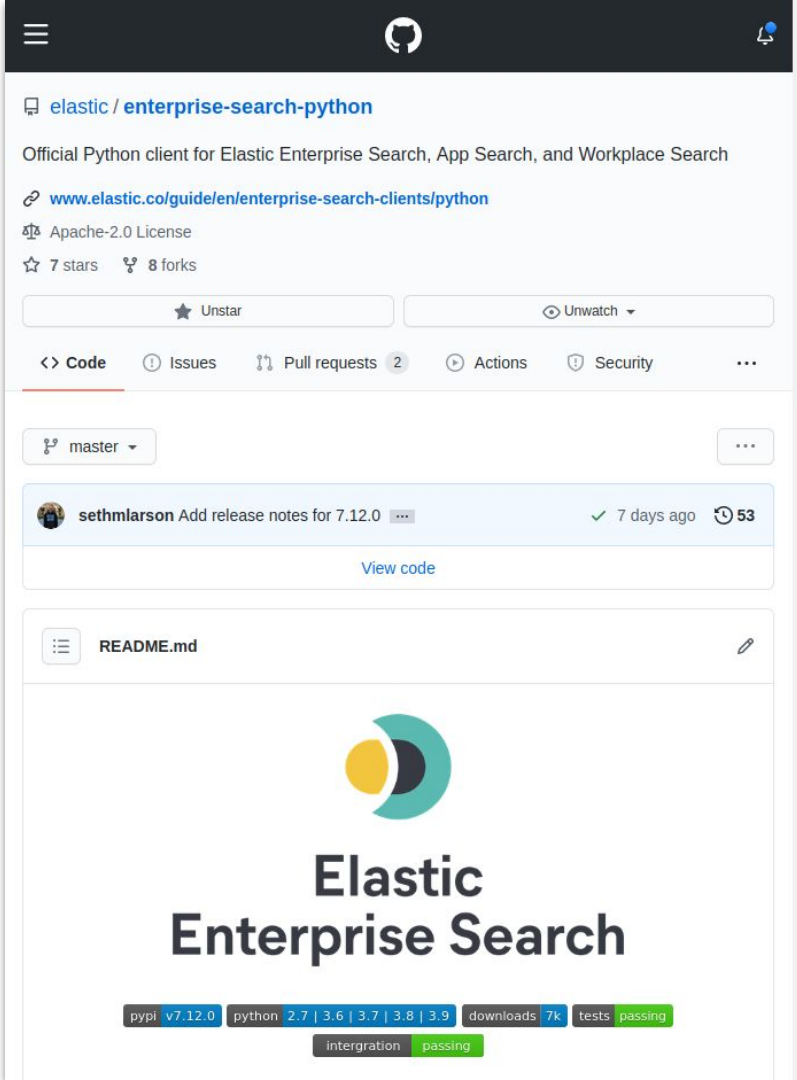
Source code available on GitHub

- `elastic/enterprise-search-python`
- Open Source, licensed Apache-2.0
- **Elastic Contributor Program**

Documentation on elastic.co/guide

- Look for “Enterprise Search Clients”

Available since Elastic Stack 7.11



How we build the Client

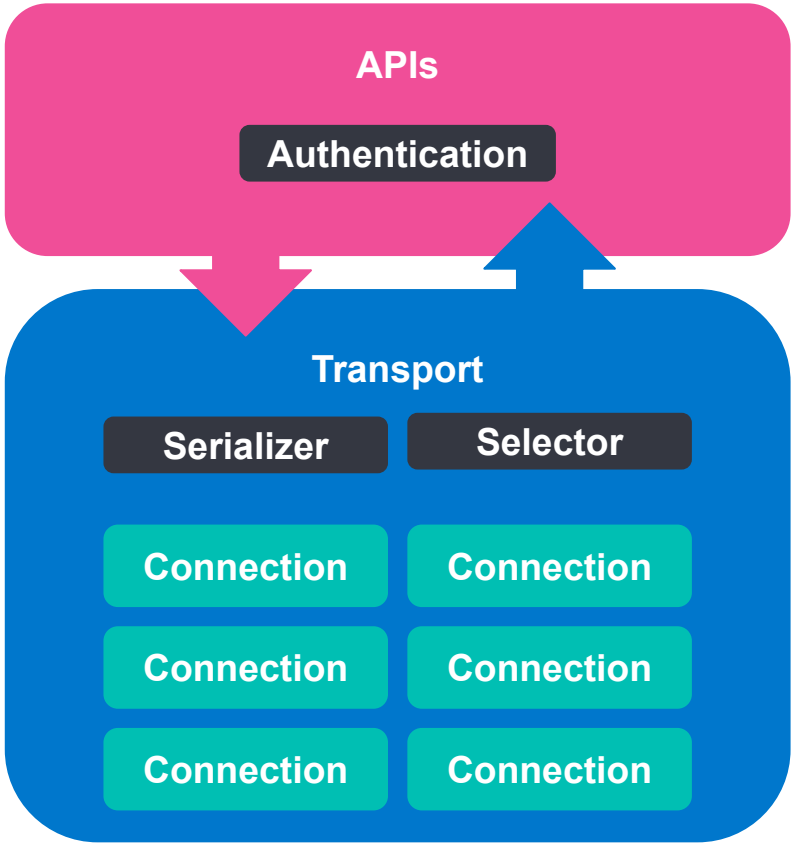
Generated from API specifications

- New APIs in App Search available on release

Client has APIs for all Enterprise Search services

Transport Layer is from Elasticsearch client

- Battle-tested by millions of users
- Reusable component for future Elastic clients



elastic

Getting our hands dirty!

Elastic Cloud

14 day free trial @ cloud.elastic.co

- Spin up Elasticsearch and Enterprise Search instances
- Available in multiple cloud providers and regions
- Managed instances, upgrades, and resources

Self-Managed

Download from elastic.co/downloads

- Available as a Docker image or installers for Debian, RPM, and macOS
- Requires an Elasticsearch cluster too!
- Distributed for free via the Elastic License

Creating an Elastic Cloud deployment

This section in Elastic Cloud UI
Corresponds to Example #1 (Making Requests)

- Click ‘Create Deployment’
- Select the Enterprise Search template
- Choose your Cloud provider and region
- Mention advanced settings for more control
- Click deploy

elastic

Making Requests

This section in App Search UI
Corresponds to Example #1 (Making Requests)

- How to access Enterprise Search from Elastic Cloud deployment
- Mention Workplace Search, for now we focus on App Search
- Show off the Enterprise Search UI -> App Search
- Main App Search screen

elastic

Find App Search Credentials

This section in App Search UI
Corresponds to Example #2 (First Engine and Documents)

- Click the Credentials section
- Explain the difference between Private key and Search key
- For Python since it’s a backend language, private key is what we want
- Copy the Private key, paste into the config.yml “private_key” field
- Run the example to create an engine and index documents
    - but we’ll cover how to do this from the UI as well

elastic

Creating our first Engine

This section in App Search UI
Corresponds to Example #2 (First Engine and Documents)

- Click the Engines section
- Create Engine
- Pick a name, “national-parks-pycon-2021”
- Language if you know, otherwise set to Universal, our data is in English
- An empty Engine has been created!

elastic

Indexing Documents

This section in App Search UI
Corresponds to Example #2 (First Engine and Documents)

- With our empty engine open
- Bunch of ways to index documents
  - Via the API in the example
  - Paste JSON
  - Upload JSON file
  - Web Crawler (we'll talk about this later!)
- Go the route of Upload JSON file (data.json)
- See that our documents are indexed

elastic

Refine the Engine Schema

This section in App Search UI
Corresponds to Example #3 (Refine Schema)

- Our engine has documents, now let's check out the schema
- Schema is how App Search indexes each field
- Default is text for all fields
- Four types are text, number, date, and geolocation
    - Text is for strings small and large, enums
    - Number is for integers or floats
    - Date is for dates or datetimes
    - Geolocation is for points only
    - No boolean type, booleans should be text "true" or "false"
- *Set all the fields to their proper types*
- App Search automatically re-indexes your data for you, does the switch with no downtime
- Also alerts you of problems with documents after reindexing

elastic

Indexing Django Models

elastic

Indexing Django Models into App Search

Model Signals

Keep models in App Search up-to-date with Signal receivers

- Signals of interest: `post_save`, `post_delete`, `m2m_changed`
- ForeignKeys and ManyToManyFields require more work to denormalize
- **Pro:** Updates available as soon as they reach your database
- **Con:** No synchronization mechanism in case of failures or desync

Indexing Task and Meta Engines

Regularly query database for all models and index into App Search

- Meta Engines as an alias
- Source Engines for schema and documents
- **Pro:** Less complex, easier to reason about denormalization
- **Con:** Updates wait until next task execution

elastic

Preparing Django Models for App Search

Most scalar fields map to App Search nicely

- IntegerField, FloatField, DecimalField → number
- TextField, CharField → text
- DateField, DatetimeField → datetime

App Search ID

- Derive from or store with Model

Boolean fields

- No boolean type in App Search
- Map to App Search text as an “enum”

App Search geolocation

- Latitude + Longitude as fields → “<lat>, <long>”
- Recommend DecimalField for precision

```
# Django
class Park(models.Model):
    id = models.CharField(primary_key=True)
    title = models.TextField()
    latitude = models.DecimalField(decimal_places=4)
    longitude = models.DecimalField(decimal_places=4)
    area = models.FloatField()
    established = models.DateTimeField()
    world_heritage_site = models.BooleanField()
    visitors = models.IntegerField()

# App Search
{
    "id": "park-zion",
    "title": "Zion",
    "location": "37.3, -113.05",
    "area": 595.8,
    "established": "1919-11-19T06:00:00Z",
    "world_heritage_site": "false",
    "visitors": 4295127
}
```

Preparing Django Models for App Search

“One to Many” and “Many to Many” require more work

- Arrays of scalars allowed in fields of that type
- “States” field in App Search is type “text”

Objects in arrays not allowed

- Instead denormalize to multiple fields in App Search

Search focuses on content, flattened data is good!

```
# Django
class State(models.Model):
    name = models.CharField(primary_key=True)
    abbrev = models.CharField()


class Park(models.Model):
    ...
    states = models.ManyToManyField(State)


# Model in Django DB:
Park(
  states=[
    State(name="California", abbrev="CA"),
    State(name="Nevada", abbrev="NV")
  ]
)


# App Search
{
    "state_names": ["California", "Nevada"],
    "state_abbrev": ["CA", "NV"]
}
```

A note on Datetimes and Timezones

Datetime objects without timezones aren't recommended

- Don't use the `datetime.utcnow()` or `datetime.utcfromtimestamp()`
- Client assumes **local time if no timezone information**

RFC 3339 datetime format requires timezones

`YYYY-mm-ddTHH:MM:SS{Z|+MM:SS|-MM:SS}`
(example "`2021-04-01T19:20:21Z`")

"Stop using utcnow and utcfromtimestamp"
~ Paul Ganssle, Python timezone wizard

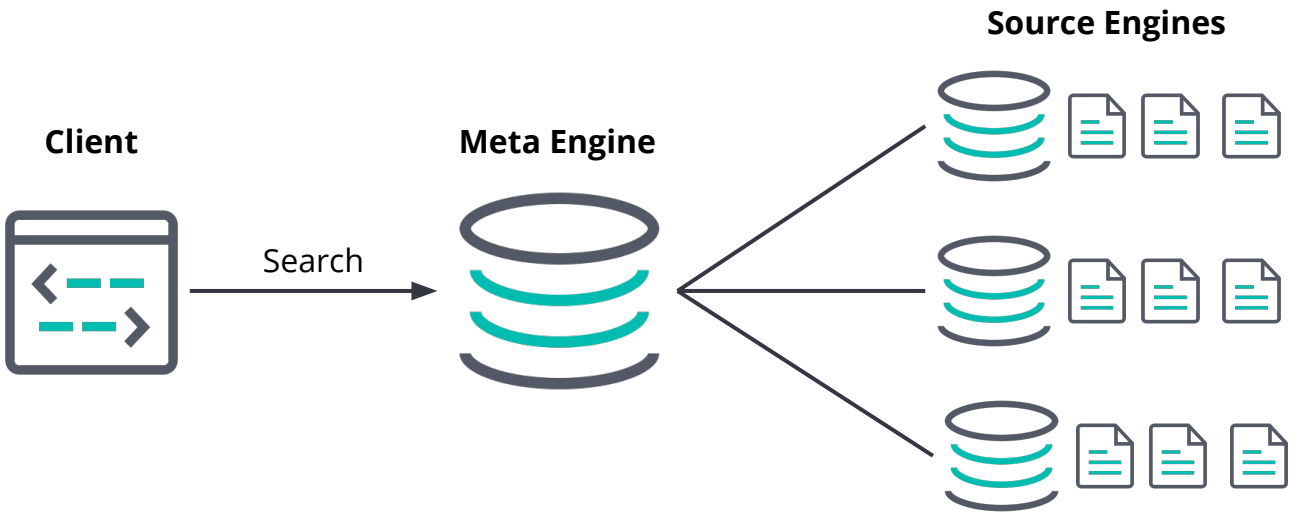
```
# DO THIS:
from datetime import datetime, timezone

dt = datetime.now(tz=timezone.utc)

# NOT THIS:
from datetime import datetime

dt = datetime.utcnow()
```

elastic

What is a Meta Engine?

Meta Engines are engines where Documents live in other engines

- Think of them like aliases for 1+ source engines
- Platinum licensed feature



elastic

Scheduled indexing with Meta Engines

Use a task scheduler like Celery / APScheduler / cron

Task should do the following:

1. Create a new Engine
2. Query the database for all models
3. Index all models into the new Engine
4. Change the Meta Engine source to the new Engine
5. Delete the old source Engine(s)

elastic

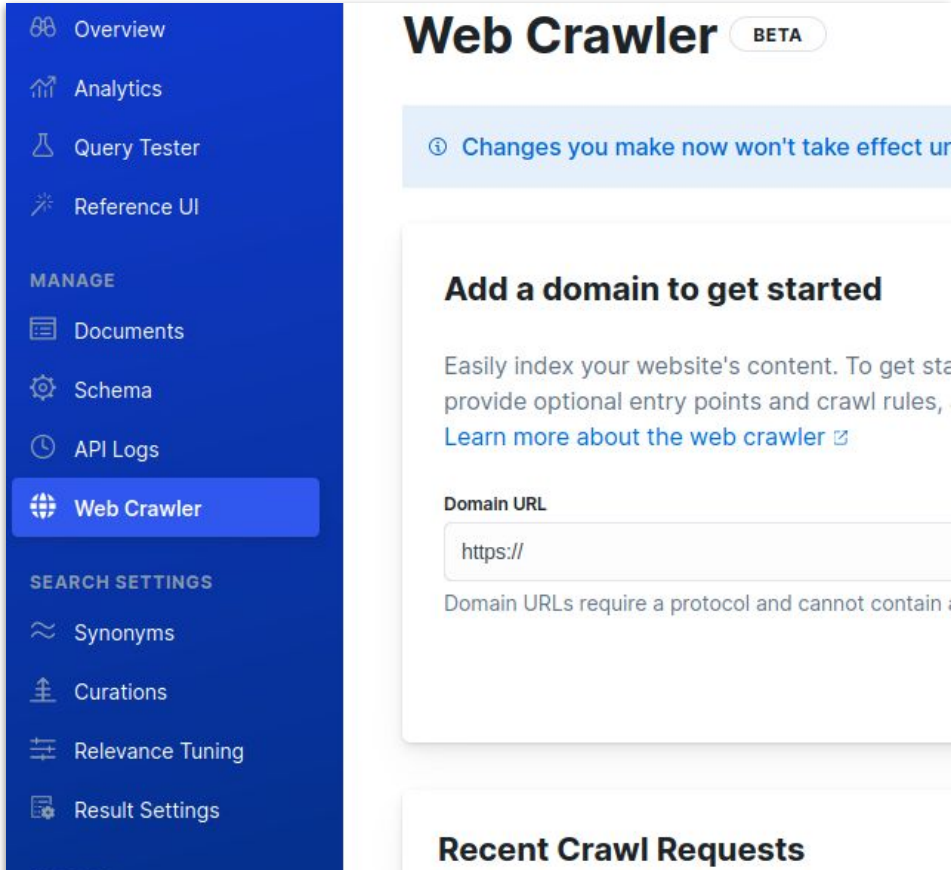
Index Documents via Web Crawler

elastic

App Search crawls your Web Applications

- Add a domain, crawler keeps the engine up to date with content from accessible pages. **You care about crawlability right?**
- Web Crawler is in **beta**
- **Pro:** Set and forget, your pages are all indexed and kept up-to-date automatically
- **Con:** Less control over exact structure of documents*. Focuses more on per-page discovery

  * **Coming soon:** Meta tags will allow more control of document structure.



What does the Web Crawler need from Django?

Figure out your entry points and path patterns

- Add routes you want to the crawler to start from like `/blog` or `/store`
- Crawler follows links that match path patterns

Implement crawling + SEO best practices:

- sitemap.xml
- robots.txt
- title, meta description, nofollow

```
# robots.txt (http://www.example.com/robots.txt)

User-agent: *
Allow: /

Sitemap: http://www.example.com/sitemap.xml


# sitemap.xml (http://www.example.com/sitemap.xml)

<?xml version="1.0" encoding="UTF-8"?>
<sitemapindex xmlns="http://www.sitemaps.org/schemas/sitemap
   <sitemap>
      <loc>http://www.example.com/store/1</loc>
      <lastmod>2004-10-01T18:23:17+00:00</lastmod>
   </sitemap>
   <sitemap>
      <loc>http://www.example.com/store/2</loc>
      <lastmod>2005-01-01</lastmod>
   </sitemap>
</sitemapindex>
```

elastic

Using the Web Crawler UI

This section in App Search UI

- Open up an empty engine
- Select the Web Crawler index documents option
- Configure for elastic.co/blog as entry point
- Filter for elastic.co/blog/*
- Start the Crawler, see documents coming in
- Open an example document, try a search!

elastic

Let the Searching Commence!

elastic

Searching with the Client

✨ **`AppSearch.search()`** ✨

- `results[]` for documents
- `meta.page` for pagination
- `meta.request_id` for analytics
- Automatically sorted by score

Search options that are available

- Filtering
- Pagination
- Sorting
- Facets (Aggregations)

```
>>> client.search(
    engine_name="national-parks-pycon-2021",
    body={"query": "mountain"}
)

{
  "meta": {
    "alerts": [],
    "warnings": [],
    "page": {
      "current": 1,
      "total_pages": 3,
      "total_results": 24,
      "size": 10
    },
    "engine": {
      "name": "national-parks-pycon-2021",
      "type": "default"
    },
    "request_id": "DChdPn5iTImEbUwARLMdYQ"
  },
  "results": [
    {
      "title": {
        "raw": "Rocky Mountain"
      },
      "url": {
        "raw": "https://www.nps.gov/romo/index.htm"
      },
      "_meta": {
        "engine": "national-parks-pycon2021",
        "score": 71.24352,
        "id": "park-rocky-mountain"
      },
      "id": {
        "raw": "park-rocky-mountain"
      }
    },
```

Query Suggestions

Submit a partial query, quickly receive suggested queries with high scoring results

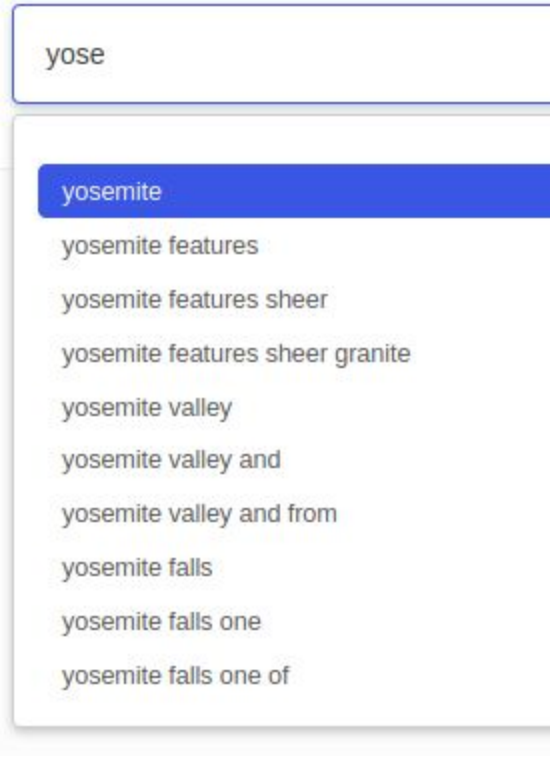
Also known as “type-ahead” or “auto-complete”

What if I want result suggestions?

- Use the Search API for result suggestions

How is Query Suggestions different from Search?

- Faster, less search logic, server-side optimizations
- Uses “`best_fields`” Elasticsearch match type
- Compared to “`cross_fields`” match type for Search



elastic

Signed Search Keys

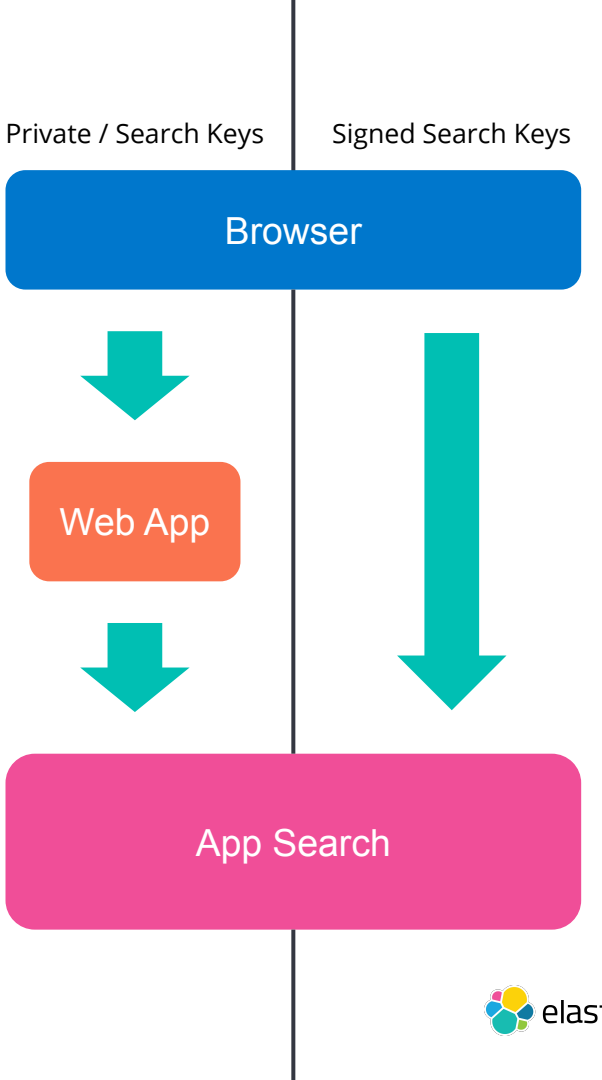
`AppSearch.create_signed_search_key()`

- Create a Bearer token for only the Search API
- Useful for client-side searching (JavaScript*)

Configure fields to search and use as results

Reduce number of requests on your web application

- Queries can be sent directly to App Search instead



* **Coming soon:** A new JavaScript client for Enterprise Search services

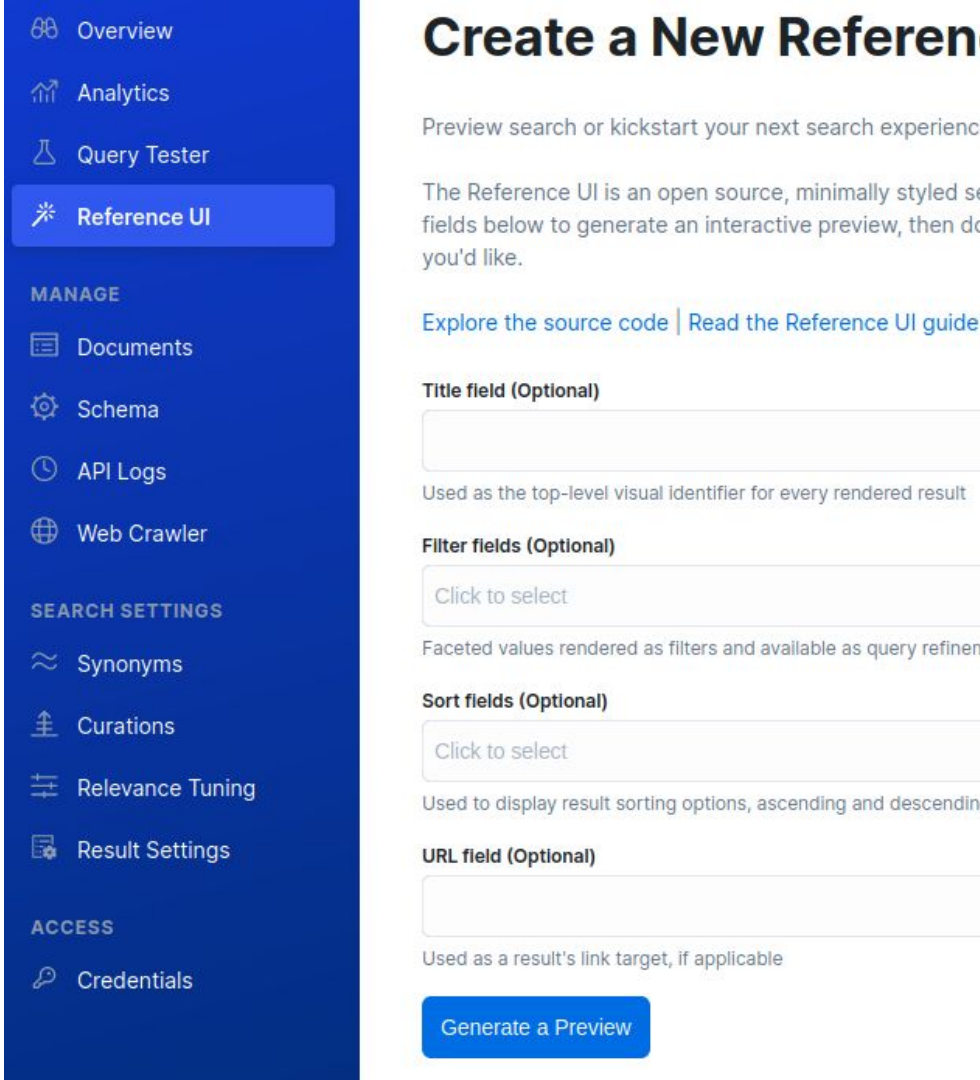
elastic

Reference UI

Not a JavaScript workshop but…

Guided creation of web search experience

- Select the fields for title, filtering, sorting
- Creates a minimal React boilerplate
- Reference UI is open source



Creating a feedback loop with Search Analytics

elastic

Analytics in App Search

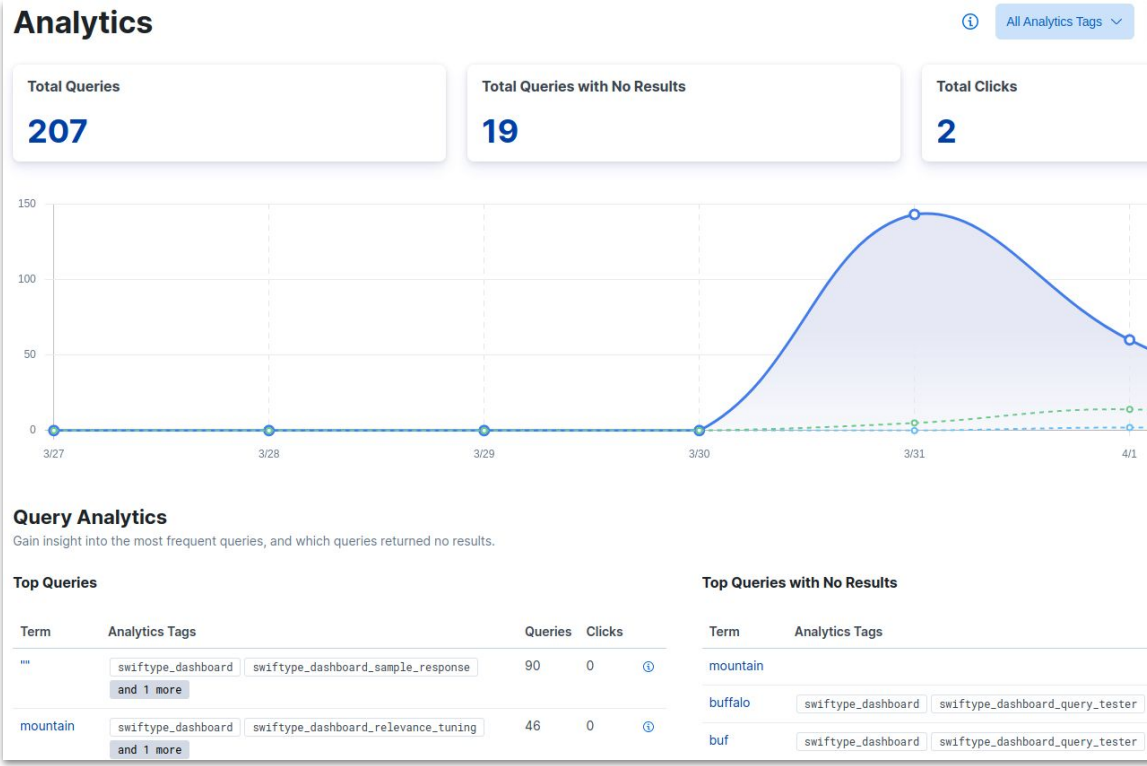
Analytics that App Search tracks

- Number of queries
- Queries per term
- Clicks per query
- Queries without clicks
- Queries without results

Queries are categorized and groupable by user-defined tags

Most analytics are automatic

- Click analytics requires one change in Python / Django

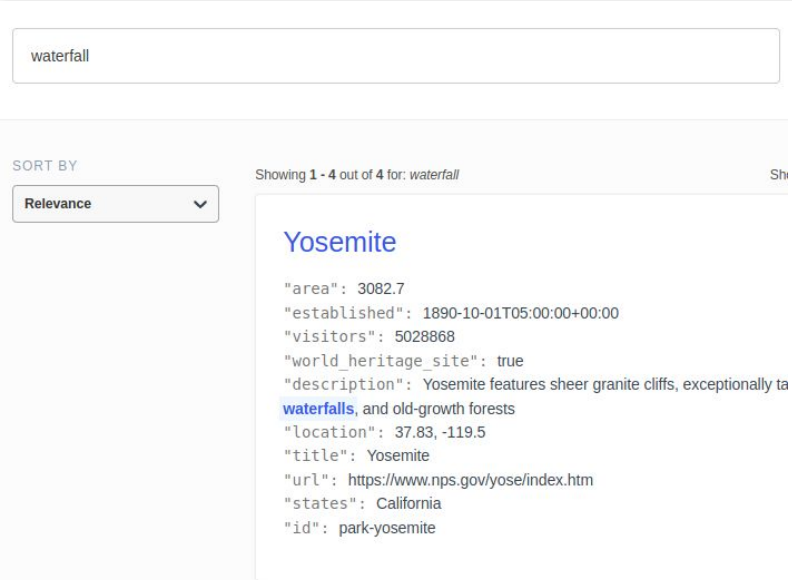


elastic

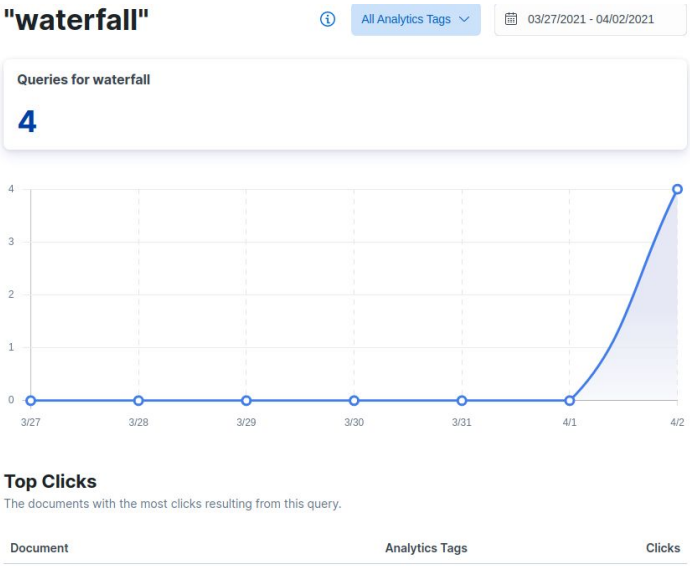
Reporting Clicks on Search Results

Save `meta.request_id` from Search result response

- Pass to `AppSearch.log_clickthrough()` along with `document_id`
- Tells App Search which documents get clicked per query



`log_clickthrough()`



Queries without Clicks or Results? ➔ Tuning!

No results or no clicks means negative search experience

- Need to use `log_clickthrough()` to detect no clicks
- Both scenarios are actionable in App Search

No results?

- **Synonym** if exact term doesn't exist in documents
- **Curation** if common phrase but not in documents

No clicks?

- **Relevance Tuning** to change ranking?
- Advice for **No Results** applies here too

elastic

Tuning your Search Results

elastic

Tuning Search Results

All the ways to tune your searches

- Relevance Tuning
- Curations
- Synonyms

Relevance Tuning

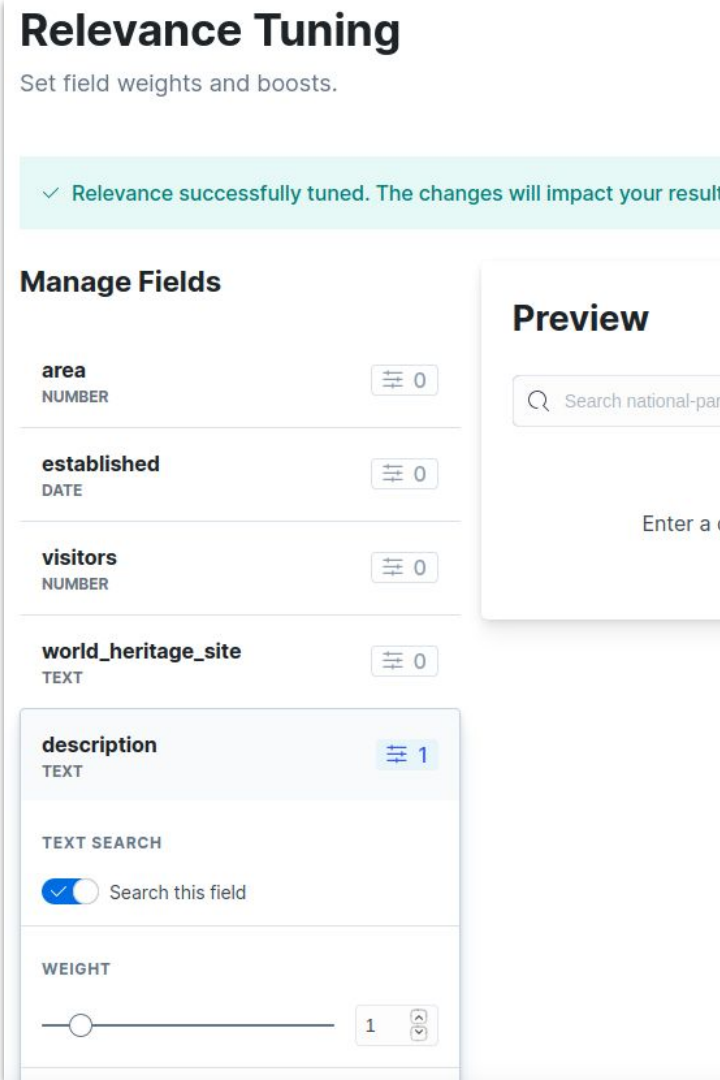
- How much individual fields contribute to document score
- Weights impact text search scoring
- Boosts impact non-text scoring

Curations

- Promote or hide documents for specific queries

Synonyms

- Words and phrases that mean the same thing



Tuning Search Results (Relevance Tuning)

This section in App Search UI
Corresponds to Example #7 (Tuning Results)

- Open the Relevance Tuning tab
- Show the live query tester and scores of documents
- Explain field weights
- Set field weights on title=3, states=2, description=1, world_heritage_site=0
- Explain boosts
- Add a log multiply boost to visitors

elastic

Tuning Search Results (Curations)

This section in App Search UI
Corresponds to Example #7 (Tuning Results)

- Start in Query Tester
- Show off “biggest park” giving a bad result
- Show the correct result “park-wrangell-st-elias”
- Open the Curations tab
- Create a new Curation for “biggest park”
- Add “park-wrangell-st-elias” to promoted
- Add “park-hot-springs” to hidden

elastic

Tuning Search Results (Synonyms)

This section in App Search UI
Corresponds to Example #7 (Tuning Results)

- Start in Query Tester
- Search for buffalo, see that there are only a few results
- Add a synonym for buffalo -> bison
- Search again, more results!

elastic

Wrapping up

App Search ❤️ Python

- New client for Enterprise Search services
- Number of community projects already growing!

Batteries included

- **Curated** APIs and developer experience
- **Web Crawler** for one-click indexing
- **Search** is simple and powerful
- **Analytics** for insights into search
- **Tuning** to maintain positive search UX

Demo Time

Elastic is a **search company.**

Search. Observe. Protect.

Elastic Technology

3 solutions

- Elastic Enterprise Search
- Elastic Observability
- Elastic Security

Powered by the Elastic Stack

- Kibana
- Elasticsearch
- Beats
- Logstash

Deployed anywhere

- **Elastic Cloud**
- **Elastic Cloud Enterprise**
- **Elastic Cloud on Kubernetes**

SaaS

Orchestration

elastic

elastic

Thank You

Elastic is a Search Company.
www.elastic.co

Join the Elastic community

1

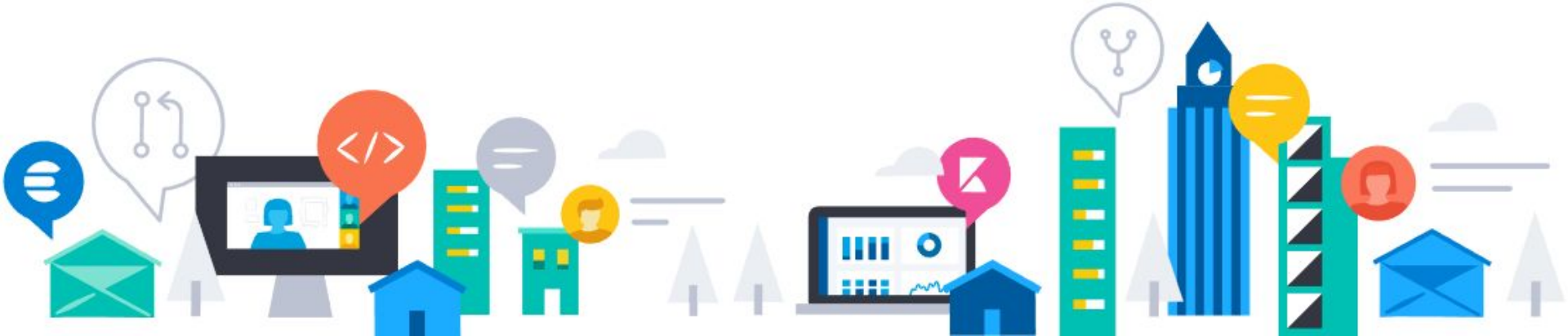
Take a quick spin:
demo.elastic.co

2

Try free on Cloud:
elastic.co/cloud

3

Connect on Slack:
ela.st/slack



elastic

Questions?

3 solutions

Elastic Enterprise Search

Elastic Observability

Elastic Security

elastic

3 solutions powered by 1 stack

Elastic Enterprise Search

Elastic Observability

Elastic Security

Elastic Stack

elastic

Deploy anywhere.

3 solutions

Elastic Enterprise Search

Elastic Observability

Elastic Security

Powered by the stack

Kibana

Elasticsearch

Beats

Logstash

Deployed anywhere

Elastic Cloud

Elastic Cloud Enterprise

Elastic Cloud on Kubernetes

SaaS

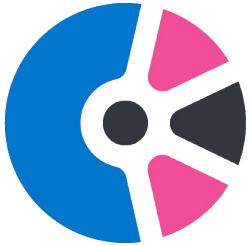
Orchestration

elastic

Deploy anywhere.

Elastic Cloud

Elastic Cloud Enterprise



Elastic Cloud on Kubernetes

SaaS

Orchestration

elastic

Elastic Enterprise Search

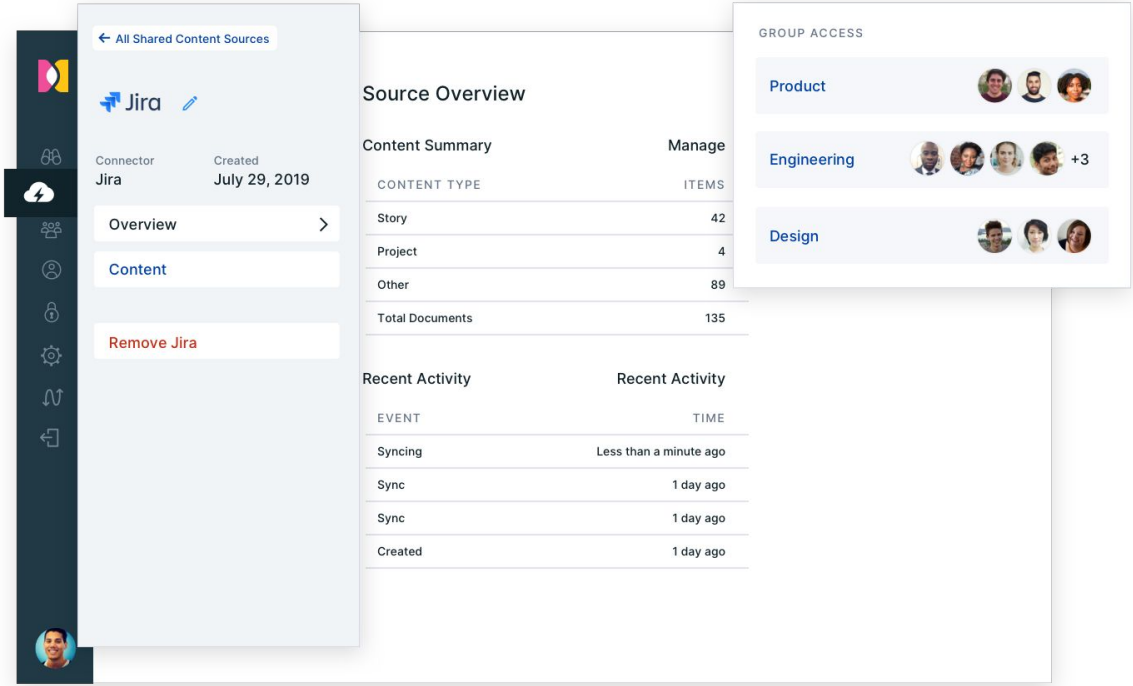
Workplace Search App Search Site Search

elastic

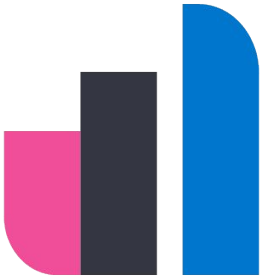
ELASTIC ENTERPRISE SEARCH

Search everything, anywhere

Easily implement powerful, modern search experiences across your website, app, or digital workplace. Search it all, simply.



elastic



Elastic Observability

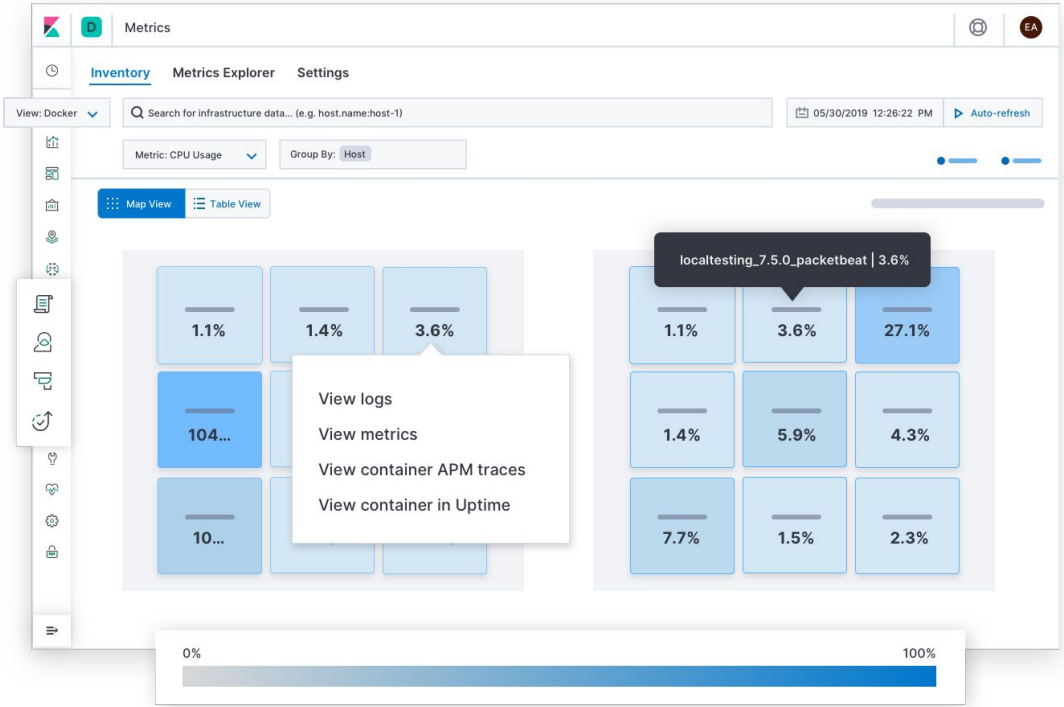
Logs Metrics APM Uptime

elastic

ELASTIC OBSERVABILITY

Unified visibility across your entire ecosystem

Bring your logs, metrics, and traces together into a single stack so you can monitor, detect, and react to events with speed.



elastic

Elastic Security

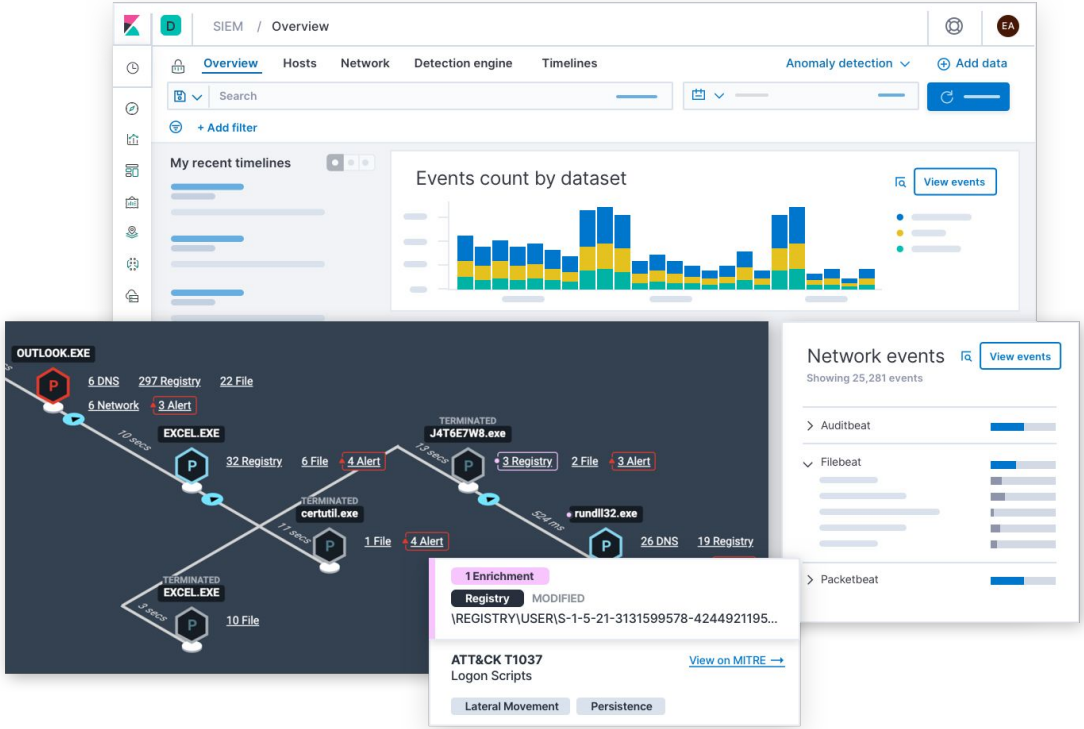
Endpoint SIEM

elastic

ELASTIC SECURITY

Security how it should be: open

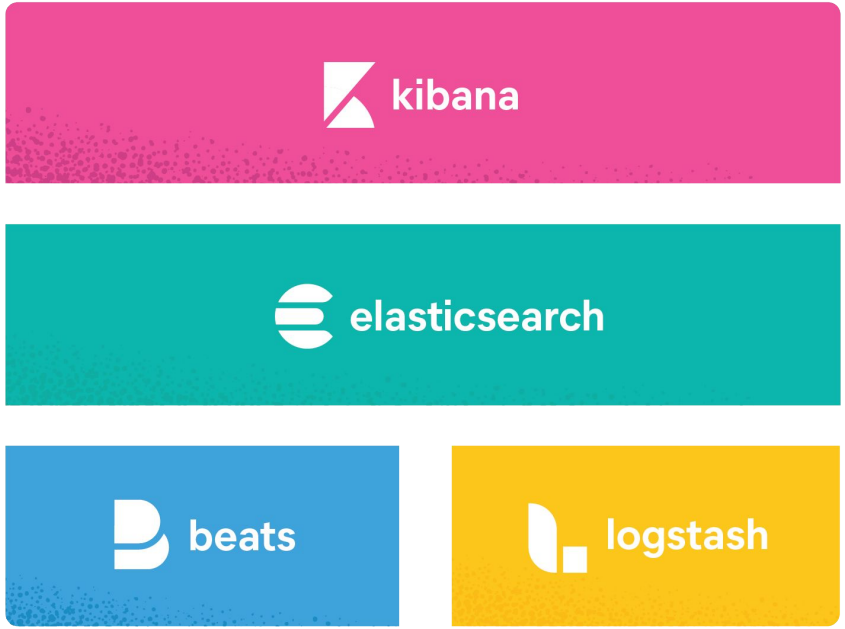
Elastic Security integrates endpoint security and SIEM to give you prevention, collection, detection, and response capabilities for unified protection across your infrastructure.



elastic

The Elastic Stack

Reliably and securely take data from any source, in any format, then search, analyze, and visualize it in real time.



elastic

Safe Harbor Statement

This presentation includes forward-looking statements that are subject to risks and uncertainties. Actual results may differ materially as a result of various risk factors included in the reports on the Forms 10-K, 10-Q, and 8-K, and in other filings we make with the SEC from time to time.Elastic undertakes no obligation to update any of these forward-looking statements.