BackstageCON

Scale the Adoption and Delivery of Backend Plugins Using the New Backstage Framework

Who are we?

Senior Software Engineer, Spotify
Enjoys product engineering driven by customer needs. ~ 2 years of Backstage development. Part-time mentor and backpacker.

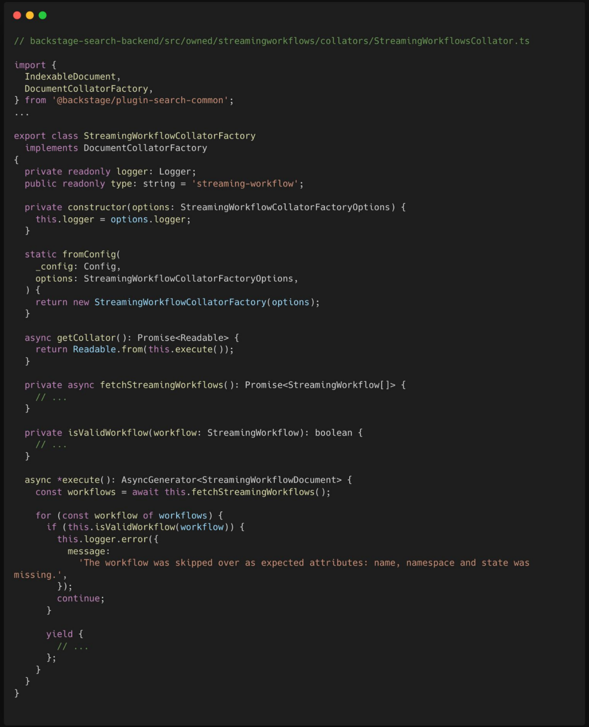
Senior Software Engineer, Spotify
5+ years of experience working on Backstage
Passionate about all things developer productivity, user experience and gardening.

Let’s go back to 2022…

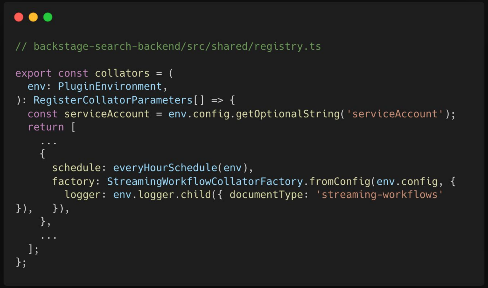
- Bring open source platform internally at Spotify
- Collaboration with multiple teams
- Figuring out ownership

… What did this look like?

Let’s go back to 2022…

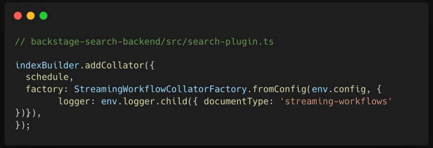


```ts
// backstage-search-backend/src/owned/streamingworkflows/collators/StreamingWorkflowsCollator.ts

import {
  IndexableDocument,
  DocumentCollatorFactory,
} from '@backstage/plugin-search-common';
...

export class StreamingWorkflowCollatorFactory
  implements DocumentCollatorFactory
{
  private readonly logger: Logger;
  public readonly type: string = 'streaming-workflow';

  private constructor(options: StreamingWorkflowCollatorFactoryOptions) {
    this.logger = options.logger;
  }

  static fromConfig(
    _config: Config,
    options: StreamingWorkflowCollatorFactoryOptions,
  ) {
    return new StreamingWorkflowCollatorFactory(options);
  }

  async getCollator(): Promise<Readable> {
    return Readable.from(this.execute());
  }

  private async fetchStreamingWorkflows(): Promise<StreamingWorkflow[]> {
    // ...
  }

  private isValidWorkflow(workflow: StreamingWorkflow): boolean {
    // ...
  }

  async *execute(): AsyncGenerator<StreamingWorkflowDocument> {
    const workflows = await this.fetchStreamingWorkflows();

    for (const workflow of workflows) {
      if (this.isValidWorkflow(workflow)) {
        this.logger.error({
          message:
            'The workflow was skipped over as expected attributes: name, namespace and state was
missing.',
        });
        continue;
      }

      yield {
        // ...
      };
    }
  }
}
```

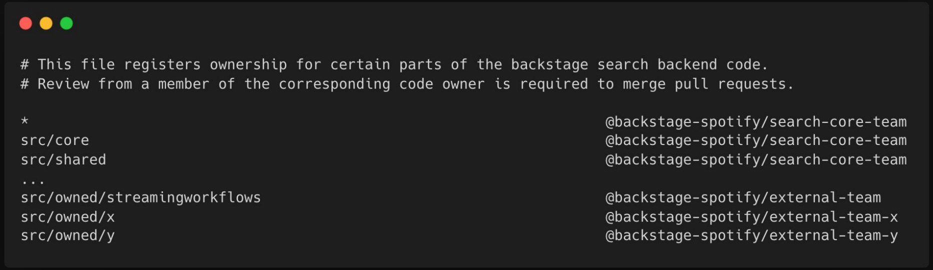


```ts
// backstage-search-backend/src/shared/registry.ts

export const collators = (
  env: PluginEnvironment,
): RegisterCollatorParameters[] => {
  const serviceAccount = env.config.getOptionalString('serviceAccount');
  return [
    ...
    {
      schedule: everyHourSchedule(env),
      factory: StreamingWorkflowCollatorFactory.fromConfig(env.config, {
        logger: env.logger.child({ documentType: 'streaming-workflows'
}),   }),
    },
    ...
  ];
};
```

…typical backend setup



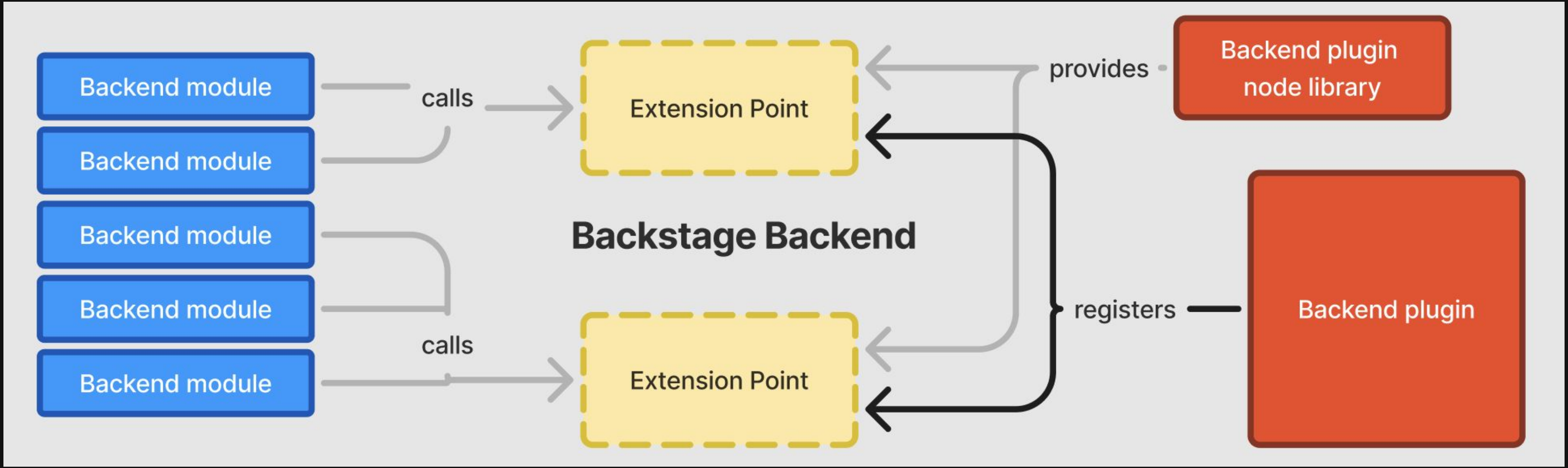
```ts
// backstage-search-backend/src/search-plugin.ts

indexBuilder.addCollator({
  schedule,
  factory: StreamingWorkflowCollatorFactory.fromConfig(env.config, {
        logger: env.logger.child({ documentType: 'streaming-workflows'
})}),
});
```



```
# This file registers ownership for certain parts of the backstage search backend code.
# Review from a member of the corresponding code owner is required to merge pull requests.

*                                                                  @backstage-spotify/search-core-team
src/core                                                           @backstage-spotify/search-core-team
src/shared                                                         @backstage-spotify/search-core-team
...
src/owned/streamingworkflows                                       @backstage-spotify/external-team
src/owned/x                                                        @backstage-spotify/external-team-x
src/owned/y                                                        @backstage-spotify/external-team-y
```

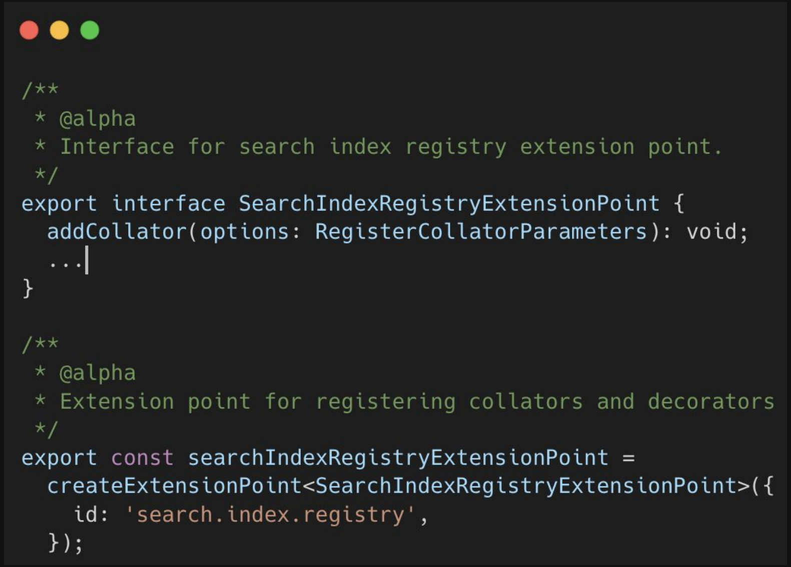
Backstage | CON

How can we solve this with the new Backend system?



Reusing search packages in the open source

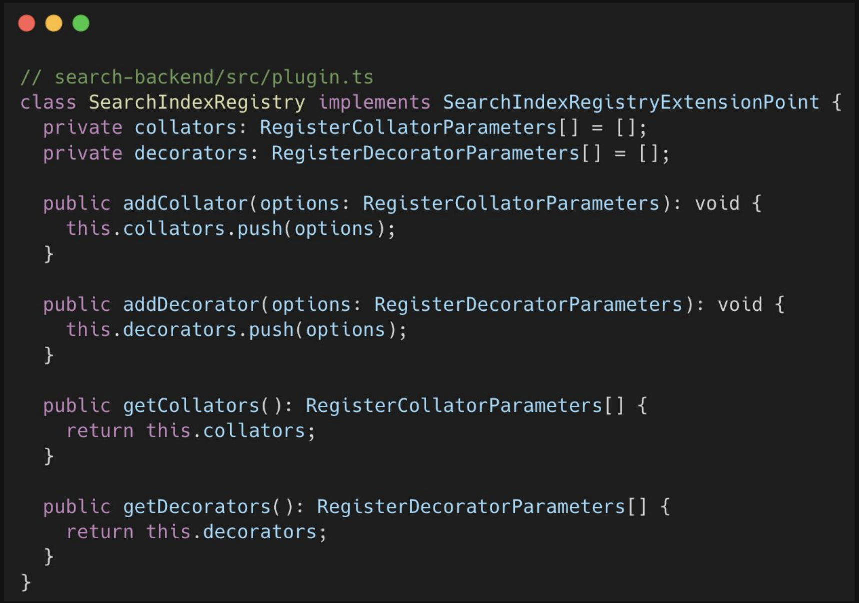
- The `searchIndexRegistryExtensionPoint` provided by the search backend node package



```
/**
 * @alpha
 * Interface for search index registry extension point.
 */
export interface SearchIndexRegistryExtensionPoint {
  addCollator(options: RegisterCollatorParameters): void;
  ...
}

/**
 * @alpha
 * Extension point for registering collators and decorators
 */
export const searchIndexRegistryExtensionPoint =
  createExtensionPoint<SearchIndexRegistryExtensionPoint>({
    id: 'search.index.registry',
  });
```

Reusing search packages in the open source

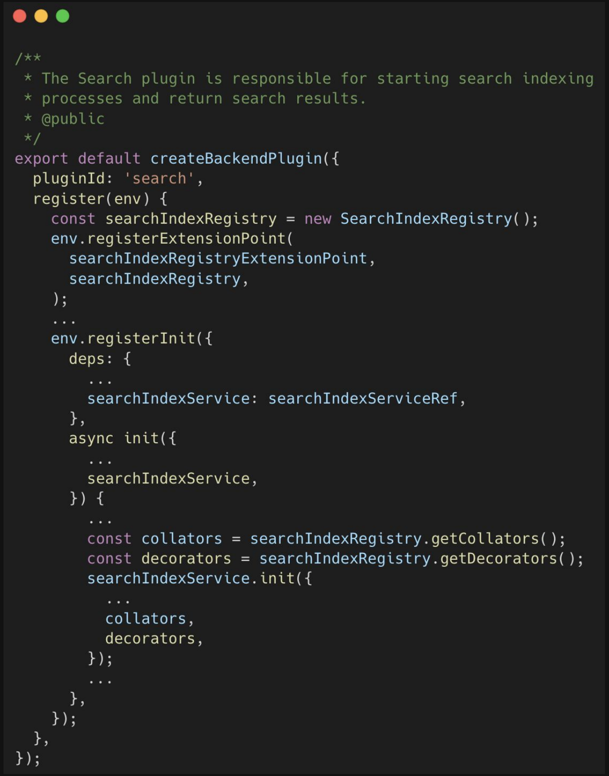
- Implementation of the `SearchIndexRegistryExtensionPoint` in the search backend plugin



```
// search-backend/src/plugin.ts
class SearchIndexRegistry implements SearchIndexRegistryExtensionPoint {
  private collators: RegisterCollatorParameters[] = [];
  private decorators: RegisterDecoratorParameters[] = [];

  public addCollator(options: RegisterCollatorParameters): void {
    this.collators.push(options);
  }

  public addDecorator(options: RegisterDecoratorParameters): void {
    this.decorators.push(options);
  }

  public getCollators(): RegisterCollatorParameters[] {
    return this.collators;
  }

  public getDecorators(): RegisterDecoratorParameters[] {
    return this.decorators;
  }
}
```

Reusing search packages in the open source

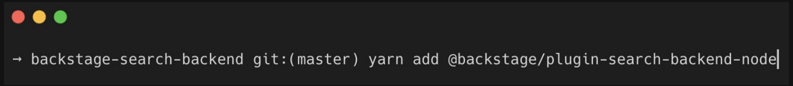
- Registering of the `searchIndexRegistryExtensionPoint` in the search backend plugin



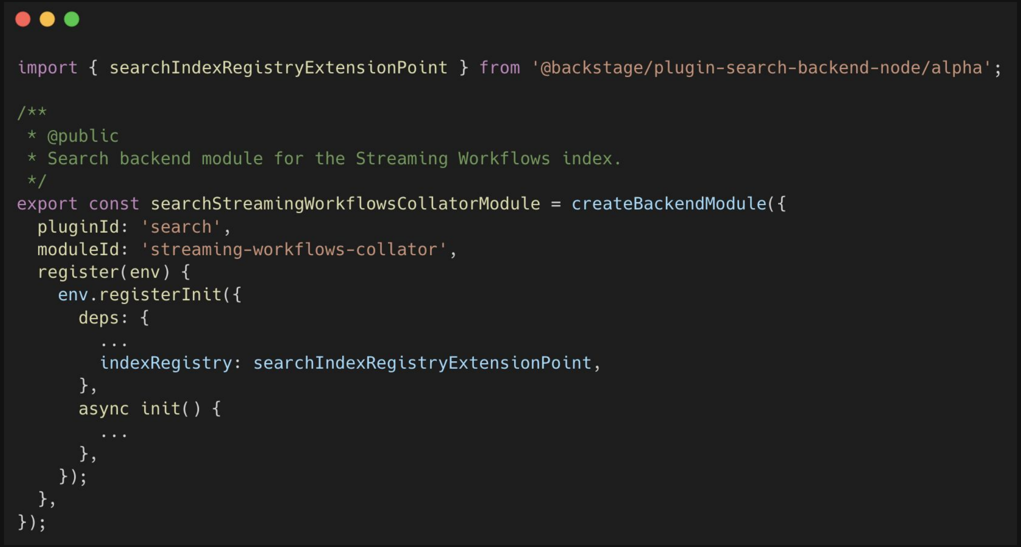
```
/**
 * The Search plugin is responsible for starting search indexing
 * processes and return search results.
 * @public
 */
export default createBackendPlugin({
  pluginId: 'search',
  register(env) {
    const searchIndexRegistry = new SearchIndexRegistry();
    env.registerExtensionPoint(
      searchIndexRegistryExtensionPoint,
      searchIndexRegistry,
    );
    ...
    env.registerInit({
      deps: {
        ...
        searchIndexService: searchIndexServiceRef,
      },
      async init({
        ...
        searchIndexService,
      }) {
        ...
        const collators = searchIndexRegistry.getCollators();
        const decorators = searchIndexRegistry.getDecorators();
        searchIndexService.init({
          ...
          collators,
          decorators,
        });
        ...
      },
    });
  },
});
```

Rewriting the internal search backend with the new Backend system

- Creating the search-backend-module-streaming-workflows-collator



```
→ backstage-search-backend git:(master) yarn add @backstage/plugin-search-backend-node
```

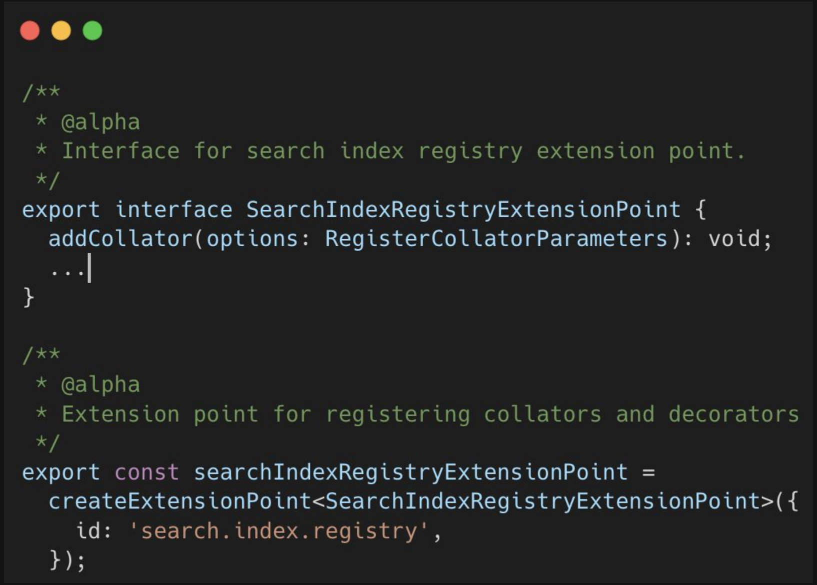


```
import { searchIndexRegistryExtensionPoint } from '@backstage/plugin-search-backend-node/alpha';

/**
 * @public
 * Search backend module for the Streaming Workflows index.
 */
export const searchStreamingWorkflowsCollatorModule = createBackendModule({
  pluginId: 'search',
  moduleId: 'streaming-workflows-collator',
  register(env) {
    env.registerInit({
      deps: {
        ...
        indexRegistry: searchIndexRegistryExtensionPoint,
      },
      async init() {
        ...
      },
    });
  },
});
```

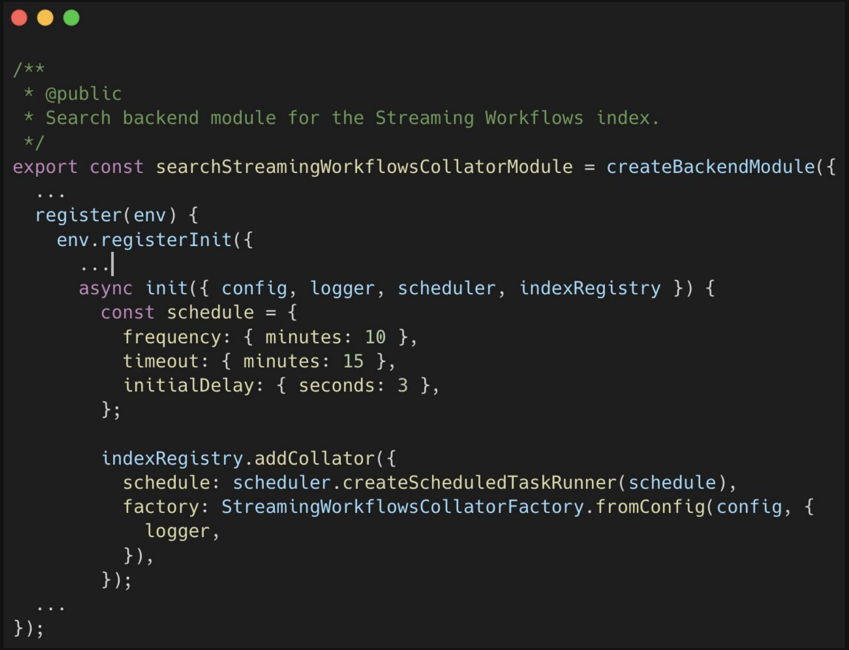
Backstage | CON

Rewriting the internal search backend with the new Backend system

- Add the streaming workflows collator to the index registry



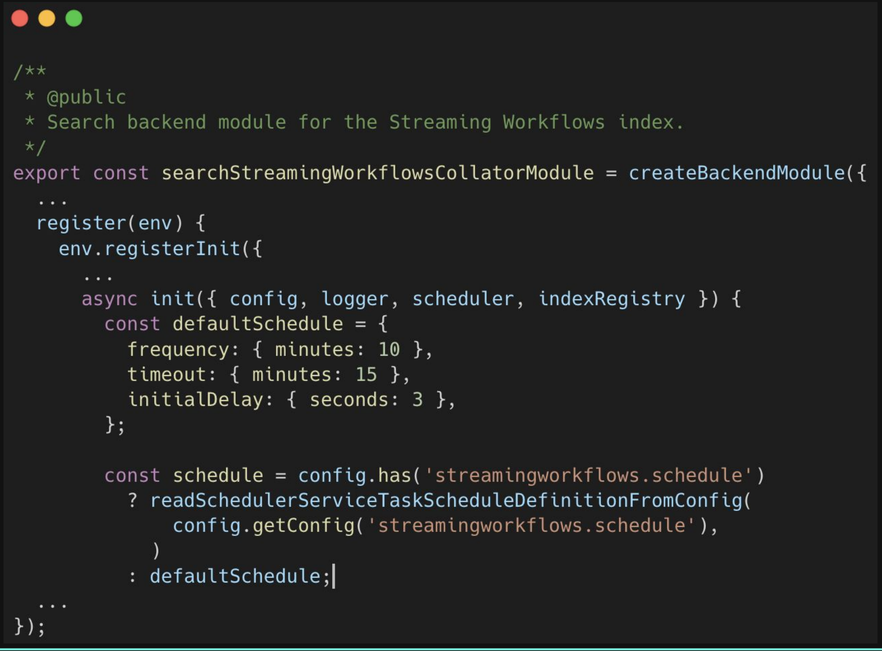
```
/**
 * @alpha
 * Interface for search index registry extension point.
 */
export interface SearchIndexRegistryExtensionPoint {
  addCollator(options: RegisterCollatorParameters): void;
  ...
}

/**
 * @alpha
 * Extension point for registering collators and decorators
 */
export const searchIndexRegistryExtensionPoint =
  createExtensionPoint<SearchIndexRegistryExtensionPoint>({
    id: 'search.index.registry',
  });
```



```
/**
 * @public
 * Search backend module for the Streaming Workflows index.
 */
export const searchStreamingWorkflowsCollatorModule = createBackendModule({
  ...
  register(env) {
    env.registerInit({
      ...
      async init({ config, logger, scheduler, indexRegistry }) {
        const schedule = {
          frequency: { minutes: 10 },
          timeout: { minutes: 15 },
          initialDelay: { seconds: 3 },
        };

        indexRegistry.addCollator({
          schedule: scheduler.createScheduledTaskRunner(schedule),
          factory: StreamingWorkflowsCollatorFactory.fromConfig(config, {
            logger,
          }),
        });
  ...
});
```

Rewriting the internal search backend with the new Backend system

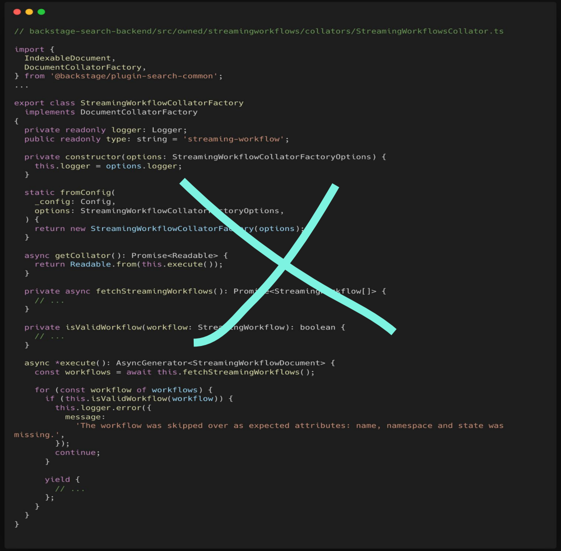
- Make the collator schedule configurable



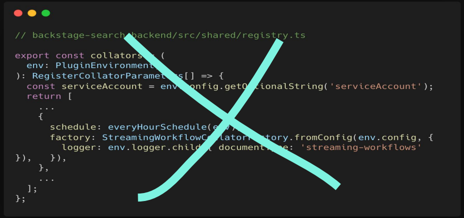
```
/**
 * @public
 * Search backend module for the Streaming Workflows index.
 */
export const searchStreamingWorkflowsCollatorModule = createBackendModule({
  ...
  register(env) {
    env.registerInit({
      ...
      async init({ config, logger, scheduler, indexRegistry }) {
        const defaultSchedule = {
          frequency: { minutes: 10 },
          timeout: { minutes: 15 },
          initialDelay: { seconds: 3 },
        };

        const schedule = config.has('streamingworkflows.schedule')
          ? readSchedulerServiceTaskScheduleDefinitionFromConfig(
              config.getConfig('streamingworkflows.schedule'),
            )
          : defaultSchedule;
  ...
});
```

Rewriting the internal search backend with the new Backend system

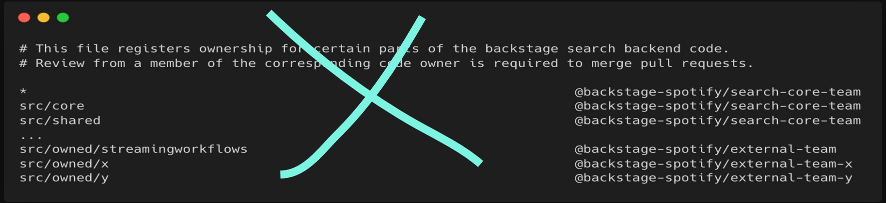
- Remove lots of code from the core search backend repo..



```
// backstage-search-backend/src/owned/streamingworkflows/collators/StreamingWorkflowsCollator.ts

import {
  IndexableDocument,
  DocumentCollatorFactory,
} from '@backstage/plugin-search-common';
...

export class StreamingWorkflowCollatorFactory
  implements DocumentCollatorFactory
{
  private readonly logger: Logger;
  public readonly type: string = 'streaming-workflow';

  private constructor(options: StreamingWorkflowCollatorFactoryOptions) {
    this.logger = options.logger;
  }

  static fromConfig(
    _config: Config,
    options: StreamingWorkflowCollatorFactoryOptions,
  ) {
    return new StreamingWorkflowCollatorFactory(options);
  }

  async getCollator(): Promise<Readable> {
    return Readable.from(this.execute());
  }

  private async fetchStreamingWorkflows(): Promise<StreamingWorkflow[]> {
    // ...
  }

  private isValidWorkflow(workflow: StreamingWorkflow): boolean {
    // ...
  }

  async *execute(): AsyncGenerator<StreamingWorkflowDocument> {
    const workflows = await this.fetchStreamingWorkflows();

    for (const workflow of workflows) {
      if (this.isValidWorkflow(workflow)) {
        this.logger.error({
          message:
            'The workflow was skipped over as expected attributes: name, namespace and state was
missing.',
        });
        continue;
      }

      yield {
        // ...
      };
    }
  }
}
```



```
// backstage-search-backend/src/shared/registry.ts

export const collators = (
  env: PluginEnvironment,
): RegisterCollatorParameters[] => {
  const serviceAccount = env.config.getOptionalString('serviceAccount');
  return [
    ...
    {
      schedule: everyHourSchedule(env),
      factory: StreamingWorkflowCollatorFactory.fromConfig(env.config, {
        logger: env.logger.child({ documentType: 'streaming-workflows'
}),   }),
    },
    ...
  ];
};
```

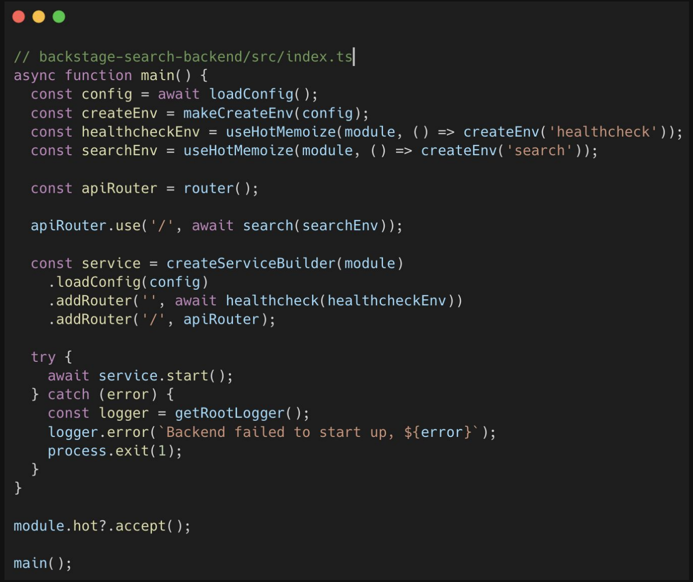


```
# This file registers ownership for certain parts of the backstage search backend code.
# Review from a member of the corresponding code owner is required to merge pull requests.

*                                                           @backstage-spotify/search-core-team
src/core                                                    @backstage-spotify/search-core-team
src/shared                                                  @backstage-spotify/search-core-team
...
src/owned/streamingworkflows                                @backstage-spotify/external-team
src/owned/x                                                 @backstage-spotify/external-team-x
src/owned/y                                                 @backstage-spotify/external-team-y
```

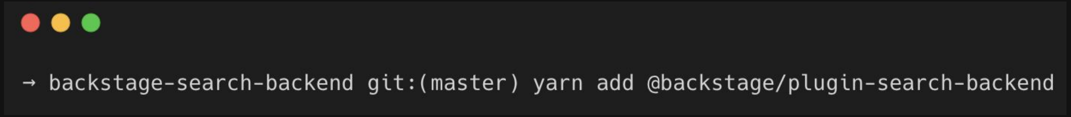
Backstage | CON

Rewriting the internal search backend with the new Backend system

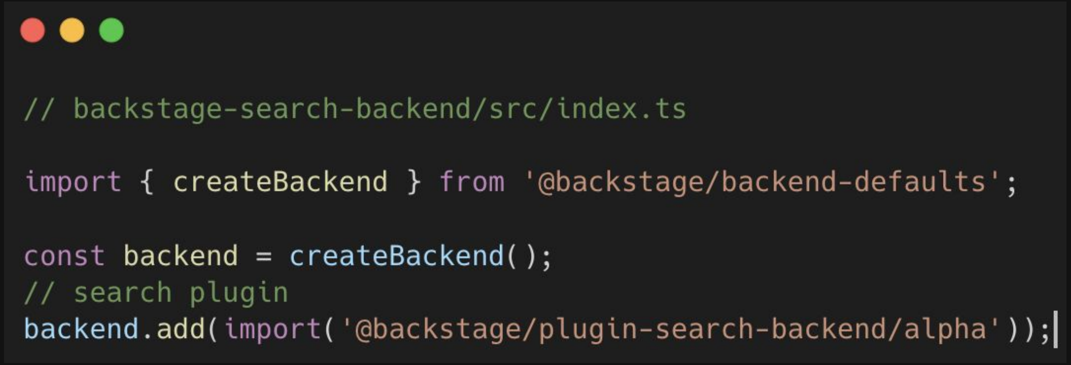
- Migrate the search backend to the new system



```ts
// backstage-search-backend/src/index.ts
async function main() {
  const config = await loadConfig();
  const createEnv = makeCreateEnv(config);
  const healthcheckEnv = useHotMemoize(module, () => createEnv('healthcheck'));
  const searchEnv = useHotMemoize(module, () => createEnv('search'));

  const apiRouter = router();

  apiRouter.use('/', await search(searchEnv));

  const service = createServiceBuilder(module)
    .loadConfig(config)
    .addRouter('', await healthcheck(healthcheckEnv))
    .addRouter('/', apiRouter);

  try {
    await service.start();
  } catch (error) {
    const logger = getRootLogger();
    logger.error(`Backend failed to start up, ${error}`);
    process.exit(1);
  }
}

module.hot?.accept();

main();
```



```
→ backstage-search-backend git:(master) yarn add @backstage/plugin-search-backend
```



```ts
// backstage-search-backend/src/index.ts

import { createBackend } from '@backstage/backend-defaults';

const backend = createBackend();
// search plugin
backend.add(import('@backstage/plugin-search-backend/alpha'));
```

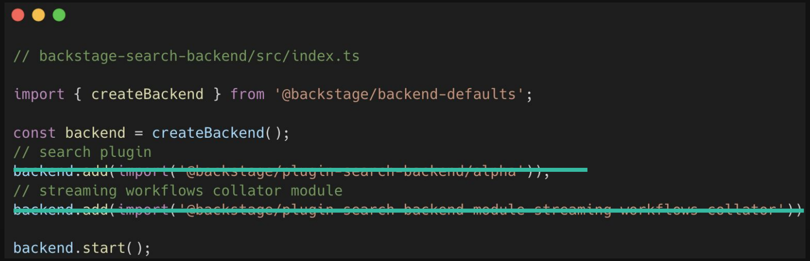
Installing the module on the internal search backend

```
➜  backstage-search-backend git:(master) yarn add @backstage-spotify/plugin-search-backend-module-
streaming-workflows-collator
```

```
// backstage-search-backend/src/index.ts

import { createBackend } from '@backstage/backend-defaults';

const backend = createBackend();
// search plugin
backend.add(import('@backstage/plugin-search-backend/alpha'));
// streaming workflows collator module
backend.add(import('@backstage/plugin-search-backend-module-streaming-workflows-collator'))

backend.start();
```

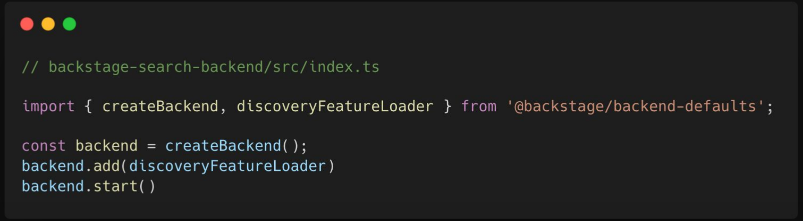
Simplifying the installation with the discovery feature loader

- Remove the module



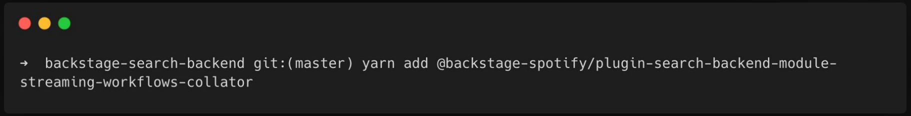
```
// backstage-search-backend/src/index.ts

import { createBackend } from '@backstage/backend-defaults';

const backend = createBackend();
// search plugin
~~backend.add(import('@backstage/plugin-search-backend/alpha'));~~
// streaming workflows collator module
~~backend.add(import('@backstage/plugin-search-backend-module-streaming-workflows-collator'))~~

backend.start();
```

- Add the discovery feature loader



```
// backstage-search-backend/src/index.ts

import { createBackend, discoveryFeatureLoader } from '@backstage/backend-defaults';

const backend = createBackend();
backend.add(discoveryFeatureLoader)
backend.start()
```

- Add the module to package.json



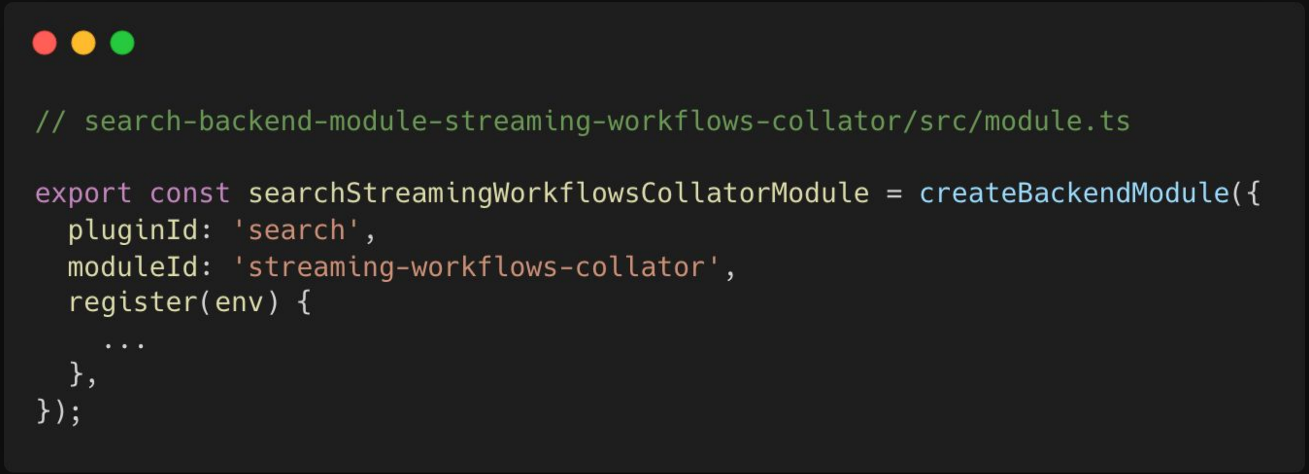
```
➜  backstage-search-backend git:(master) yarn add @backstage-spotify/plugin-search-backend-module-
streaming-workflows-collator
```

Backstage | CON

Metadata

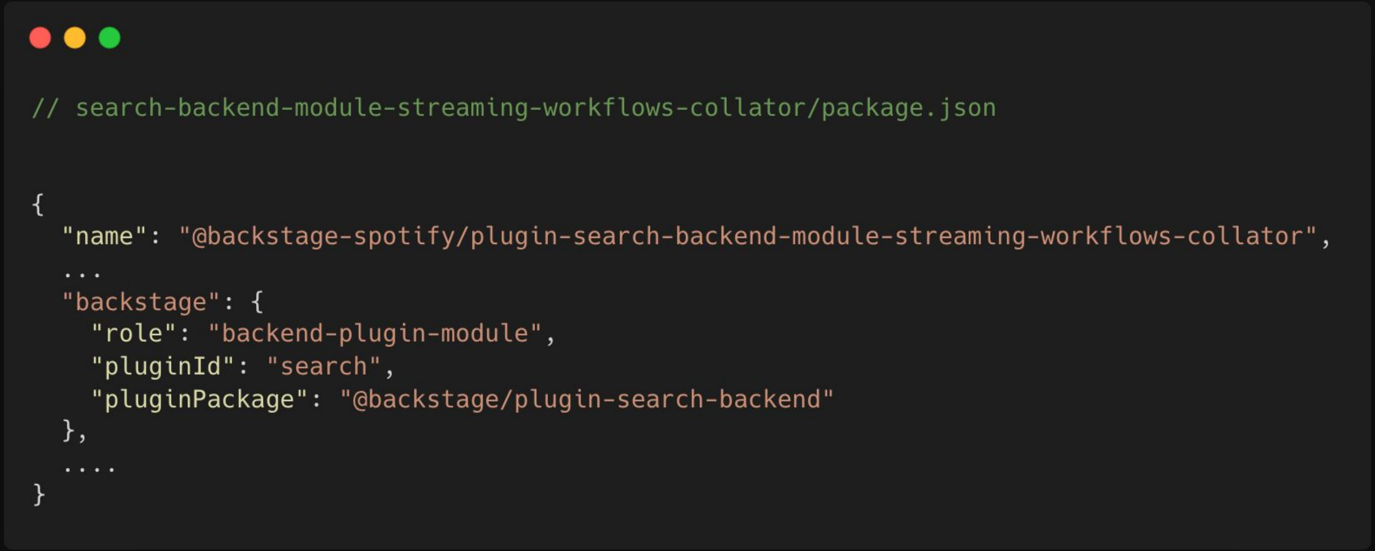
- Plugin metadata vs package metadata
- Backstage specific package metadata
- Discoverability of modules

Plugin metadata



```
// search-backend-module-streaming-workflows-collator/src/module.ts

export const searchStreamingWorkflowsCollatorModule = createBackendModule({
  pluginId: 'search',
  moduleId: 'streaming-workflows-collator',
  register(env) {
    ...
  },
});
```

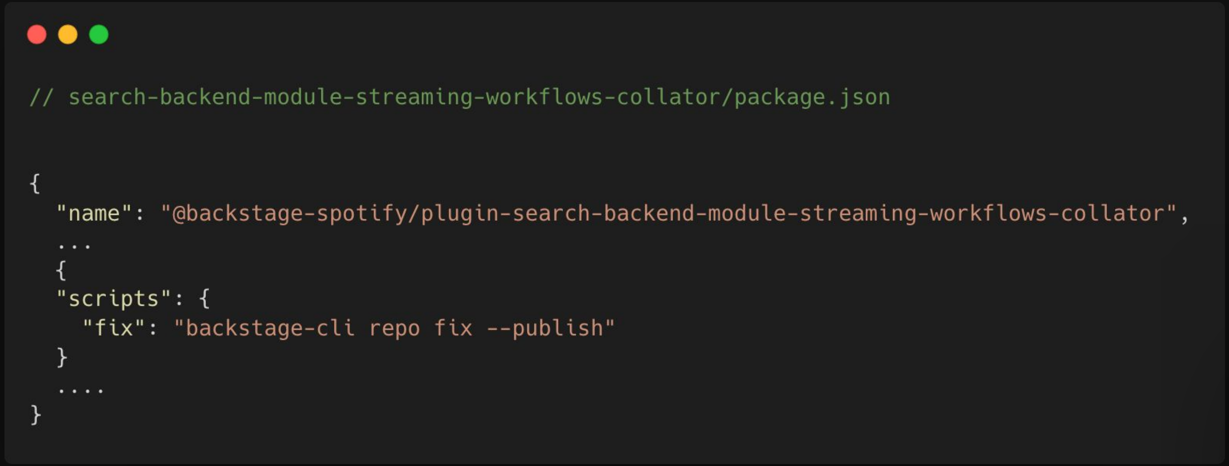
Package metadata



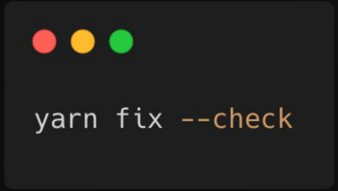
```
// search-backend-module-streaming-workflows-collator/package.json


{
  "name": "@backstage-spotify/plugin-search-backend-module-streaming-workflows-collator",
  ...
  "backstage": {
    "role": "backend-plugin-module",
    "pluginId": "search",
    "pluginPackage": "@backstage/plugin-search-backend"
  },
  ....
}
```

Package metadata

- Tooling and Automation



```
// search-backend-module-streaming-workflows-collator/package.json


{
  "name": "@backstage-spotify/plugin-search-backend-module-streaming-workflows-collator",
  ...
  {
  "scripts": {
    "fix": "backstage-cli repo fix --publish"
  }
  ....
}
```



```
yarn fix --check
```

Let’s fast forward to 202X…

NOTE: THIS WAS A HACK PROJECT. **FOR INSPIRATION ONLY.**

Benefits

Distributed ownership

Separation of Concerns

Fewer lines of code

DELIVER FASTER

Lower maintenance cost

Contained configurability

Automation and tooling support

Benefits

As-need basis customisation

Faster setup

IMPROVE ADOPTION

Accelerated time-to-value

Configuration control

I'm sold! Where do I start? 🤔

- Upgrade to the (new) backend system (Migration Docs)
- Add package metadata to your backstage packages (Package Metadata Docs)

Backstage | CON

Thank You