Observability Day
EUROPE

Customize Your Own OpenTelemetry Collector

Pablo Baeyens, *Datadog*
Evan Bradley, *Dynatrace*

About us

Observability Day
EUROPE

1 April 2025
London, England

Evan Bradley
Sr. Software Engineer
Dynatrace

Pablo Baeyens
Sr. Software Engineer
Datadog

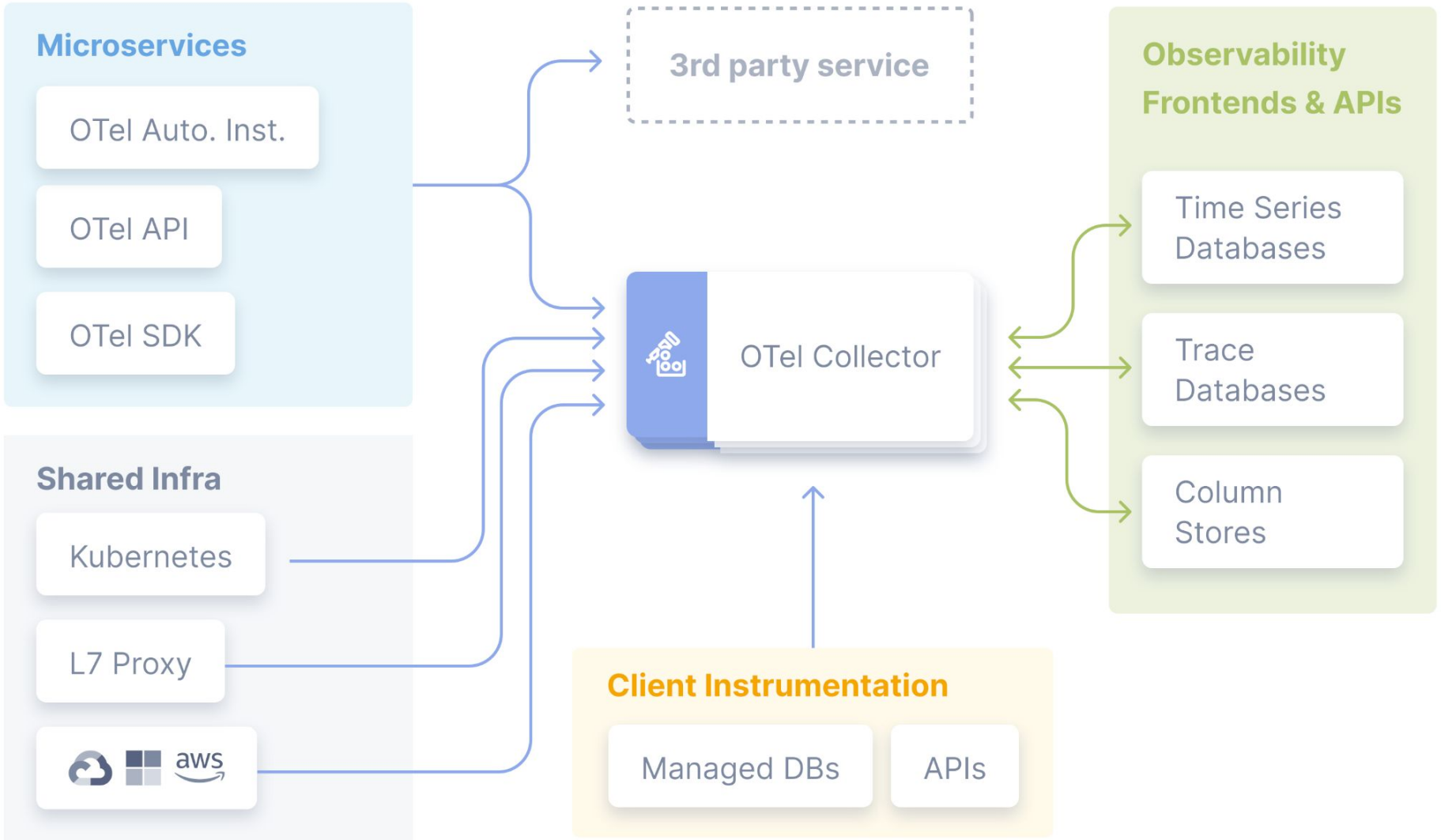
Agenda

Observability Day EUROPE

1. Intro to the Collector
   a. What is the Collector?
   b. Collector components
   c. Collector distros
2. Automating distro creation – intro to OCB
   a. Common OCB options
   b. An example distro
   c. Patching
3. Building a production-grade distro
   a. Custom components
   b. CI setup
   c. Building for your environment

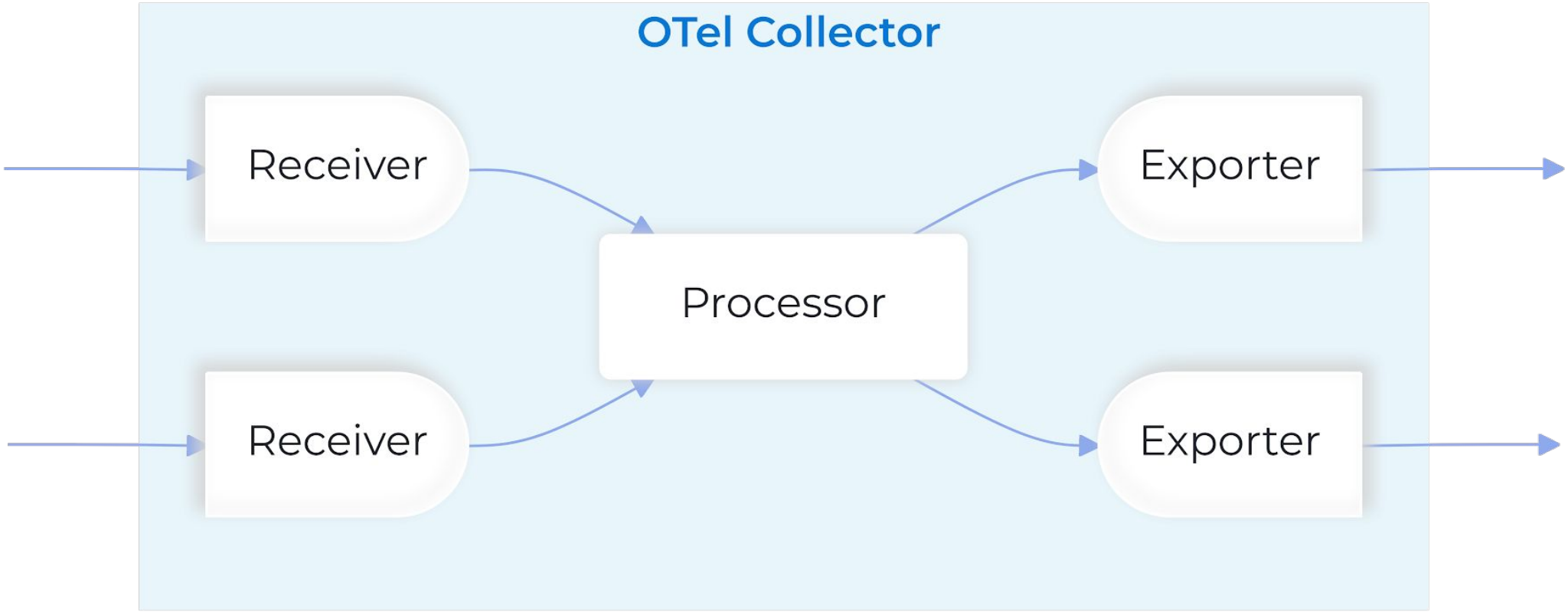
1. Intro to the Collector

What is the Collector?



OTel Collector pipelines

Observability Day
EUROPE



Collector components

A Collector consists of loosely-coupled modules, including:

Receivers

- Prometheus, Syslog...

Processors

- sampling, transform...

Extensions

- auth, health checks...

Exporters

- Debug, file...

Connectors

- spanmetrics, forward...

Config providers

- file, http, S3 buckets...

Collector distributions

A distribution (*distro*) is a binary with a certain set of components.

Upstream distros
officially released by the OTel project

Vendor distros
available from many observability vendors

Custom distros
created by users to fit their exact needs

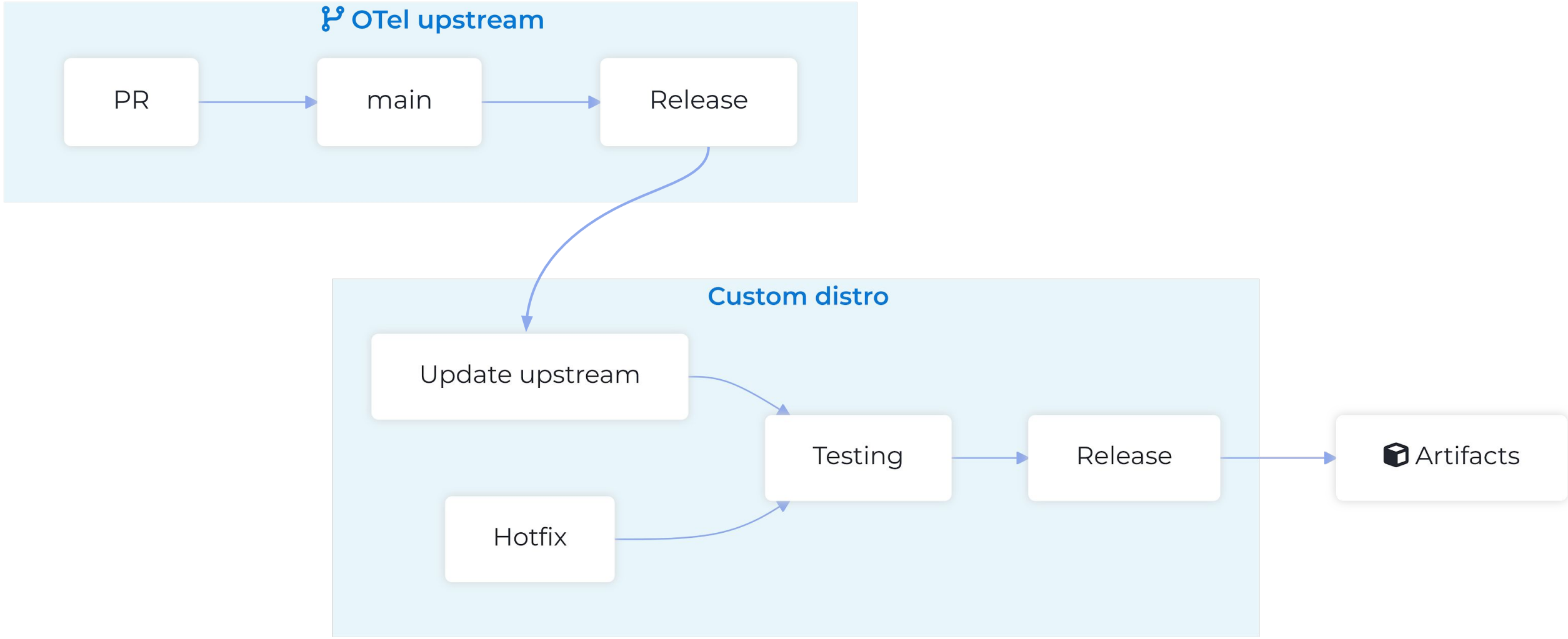
Custom distributions

Custom distros are compiled by users for their own purposes.

Everything is open source, so the sky's the limit!

- Smallest size and surface area
- Maximum flexibility
  - Pull in code early
  - Apply custom fixes
  - Add custom components
  - Customize build process

Custom distributions



Aside: *hotplugging*

What about importing components at runtime?

This:

- Is not natively supported
- Is challenging in Go
- Affects performance and security

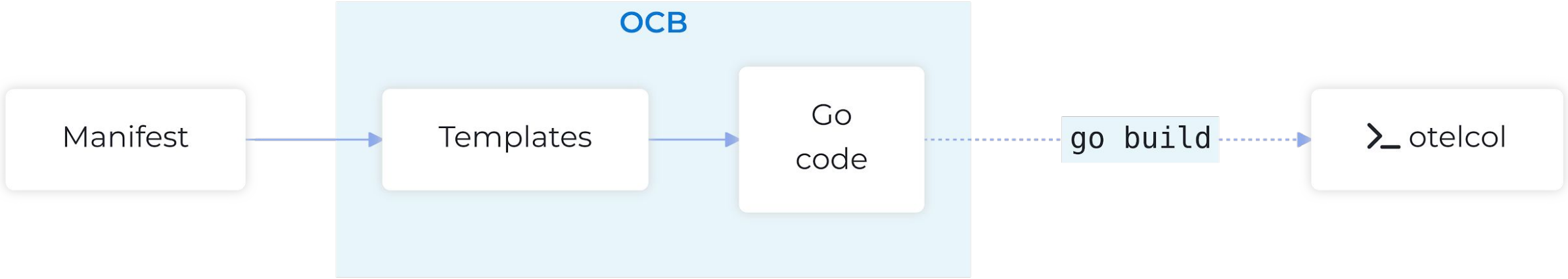
Maybe in the future?

2. The OpenTelemetry Collector Builder

Intro to OCB

The OpenTelemetry Collector Builder (OCB) automates building distros.

It takes a YAML manifest with the distro details and outputs a binary:



OCB CLI options

`--config`

- Specify the path to the OCB manifest

`--skip-compilation`

- Only generate the source files, don't compile them.
- Useful for advanced use cases.

You can see other options in the OCB documentation.

An example distro

Let’s look at how the upstream "OTLP" distro is built.
This distro can be used as a proxy (e.g. for TLS termination).

The `dist` section defines build metadata about the distro:

```
dist:
  module: github.com/open-telemetry/opentelemetry-collector-releases/otlp
  name: otelcol-otlp
  description: OpenTelemetry Collector OTLP
  version: 0.123.0
  output_path: ./bin
```

OCB manifest

Other sections pull components published as Go modules:

```
exporters:
  - gomod: go.opentelemetry.io/collector/exporter/otlpexporter v0.123.0
  - gomod: go.opentelemetry.io/collector/exporter/otlphttpexporter v0.123.0

receivers:
  - gomod: go.opentelemetry.io/collector/receiver/otlpreceiver v0.123.0

providers:
  - gomod: go.opentelemetry.io/collector/confmap/provider/envprovider v1.29.0
  - gomod: go.opentelemetry.io/collector/confmap/provider/fileprovider v1.29.0
```

API version mismatches

Each OCB version uses the matching Collector API.

- e.g. OCB v0.123.0 uses v0.123.0 beta and v1.29.0 stable.

OCB and Collector APIs may change while in beta.

- You may experience build errors if they do change and there's a mismatch.

API version mismatches

Using OCB v0.118.0 to build v0.121.0 modules fails with:

```
Error: failed to compile the OpenTelemetry Collector
distribution:
   ./components.go:30:40: undefined: extension.MakeFactoryMap
   ./components.go:41:38: undefined: receiver.MakeFactoryMap
   ./components.go:52:38: undefined: exporter.MakeFactoryMap
   ./components.go:67:40: undefined: processor.MakeFactoryMap
   ./components.go:78:40: undefined: connector.MakeFactoryMap
```

Patching upstream


```
dist:
  module: example.com/my-distro
  version: 0.123.0
  output_path: ./bin

receivers:
  - gomod: go.opentelemetry.io/[...]/otlpreceiver v0.123.0

replaces:
  - go.opentelemetry.io/[...]/otlpreceiver => hotfixes/otlpreceiver
```

3. Building a production-grade distro

Custom components (i)

A component is a Go module that exports one function:

```
func NewFactory() <kind goes here>.Factory
```

It can live anywhere, usually in the same repo as your distro.

Custom components (ii)


```
my_distro/
├─ .github/
│   ├─ workflows/
│   │   ├─ ci.yaml
├─ bin/
├─ components/
│   ├─ myreceiver/
│   │   ├─ factory.go
├─ manifest.yaml
├─ Dockerfile
```

factory.go

```
func NewFactory() receiver.Factory {
  return receiver.NewFactory(
     myReceiverType,
     defaultConfig,
     receiver.WithTraces(createTraces,...),
     receiver.WithMetrics(createMetrics,...),
     receiver.WithLogs(createLogs,...))
}
```

Custom components (iii)


```
my_distro/
├─ .github/
│  ├─ workflows/
│  │  ├─ ci.yaml
├─ bin/
├─ components/
│  ├─ myreceiver/
│  │  ├─ factory.go
├─ manifest.yaml
├─ Dockerfile
```

manifest.yaml

```yaml
dist:
  name: my-distro
  module: github.com/mycompany/otelcol
  version: 0.0.1
  output_path: ./bin

receivers:
  - gomod: example.com/myreceiver v0.0.0

replaces:
  - example.com/myreceiver => components/myreceiver
```

Building on CI


```
my_distro/
├─ .github/
│  ├─ workflows/
│  │  ├─ ci.yaml
├─ bin/
├─ components/
│  ├─ myreceiver/
│  │  ├─ factory.go
├─ manifest.yaml
├─ Dockerfile
```

ci.yaml

```yaml
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: observiq/otel-builder@v1
        with:
          manifest: "./manifest.yaml"
```

Docker images


```
my_distro/
├─ .github/
│   ├─ workflows/
│   │   ├─ ci.yaml
├─ bin/
├─ components/
│   ├─ myreceiver/
│   │   ├─ factory.go
├─ manifest.yaml
├─ Dockerfile
```

Dockerfile

```
FROM alpine:3.21 AS certs
RUN apk --update add ca-certificates # Make TLS work

FROM scratch
USER 10001:10001 # Run as non-root

COPY --from=certs /etc/ssl/certs/ /etc/ssl/certs/
COPY --chmod=755 ./bin/my-distro /my-distro
ENTRYPOINT ["/my-distro"]
```

Cross-compilation & more

Leverage the Go compiler to build for other platforms:

```
GOOS=darwin GOARCH=arm4 ocb --config […]
```

Use GoReleaser for even more! (packaging, SBOMs…)

Thank you! Questions?

Observability Day
EUROPE

OCB docs

bit.ly/otel-ocb

Download OCB

bit.ly/ocb-dl