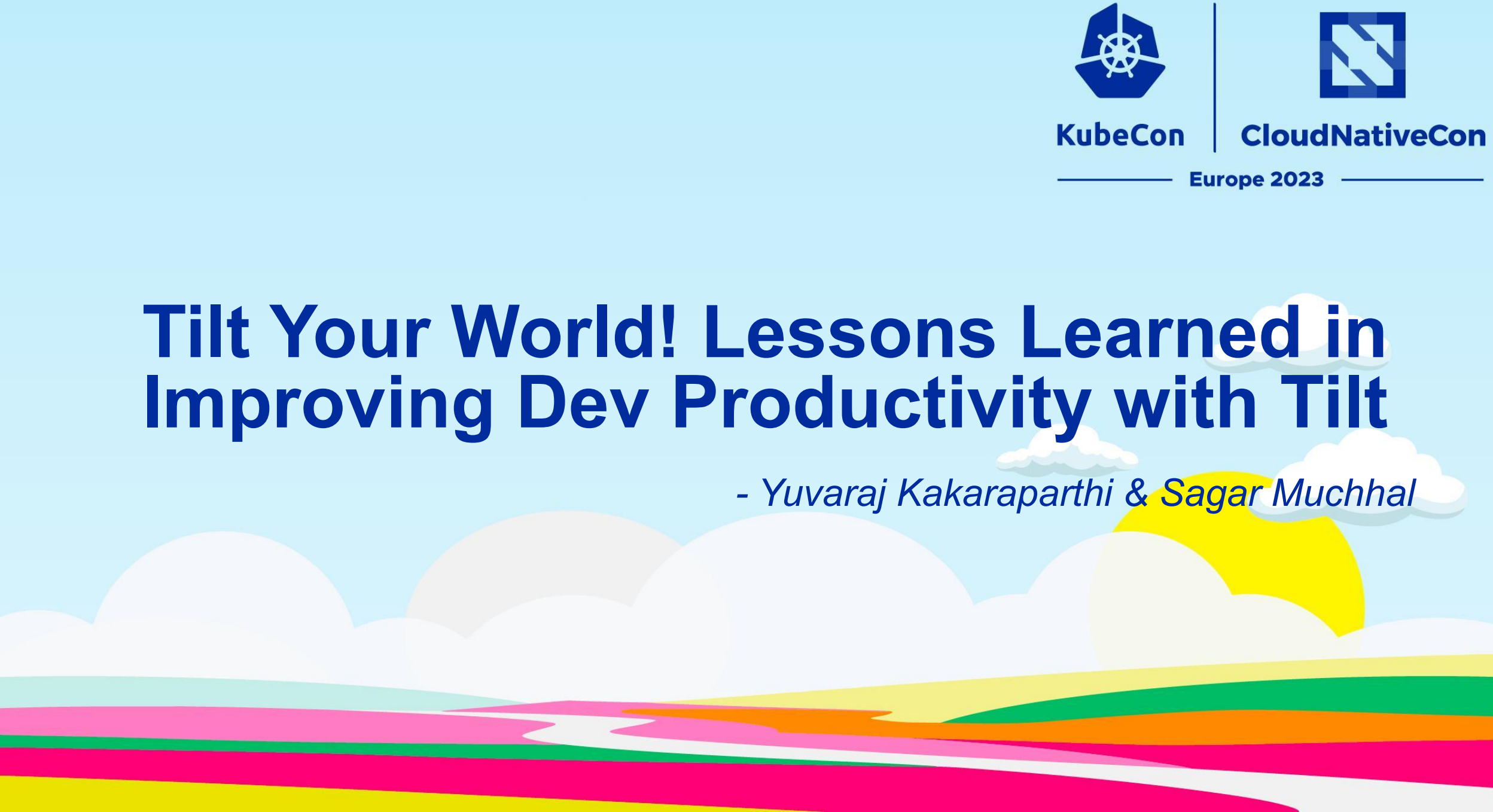

KubeCon
CloudNativeCon
Europe 2023
TiKV



Tilt Your World! Lessons Learned in Improving Dev Productivity with Tilt

- Yuvaraj Kakaraparthi & Sagar Muchhal

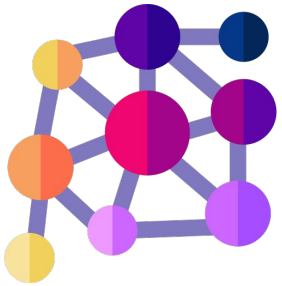
About Us

Yuvaraj Kakaraparthi
Sr MTS @VMware

Sagar Muchhal
Staff @VMware

The Problem

Modern applications are becoming increasingly distributed and complex.



*Managing applications and infrastructure is great with Kubernetes but the **development workflow is not**.*

The Problem

What makes a good development workflow?

What makes a good development workflow?

Easy spin up

Quick feedback loop

Debuggable

Visibility

What makes a good development workflow?

Easy spin up

Quick feedback loop

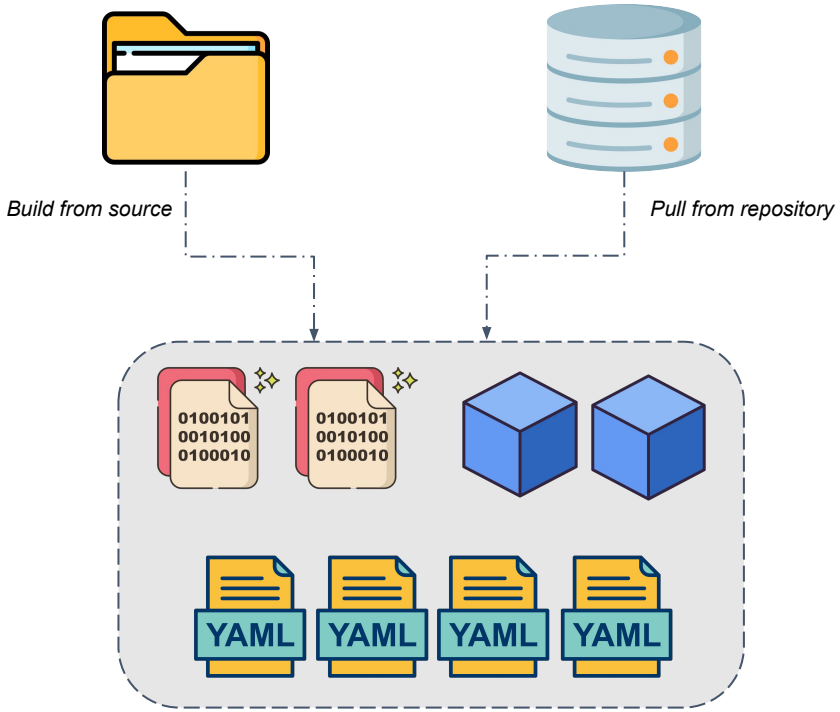
Debuggable

Visibility

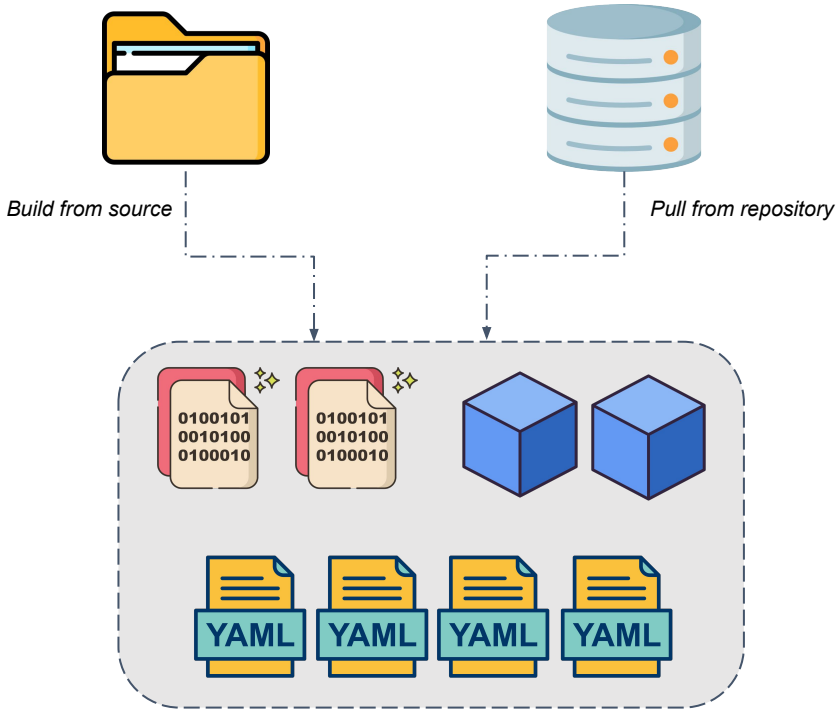
Spin up a development environment

Creating a working application environment from the desired source code.

Spin up a development environment

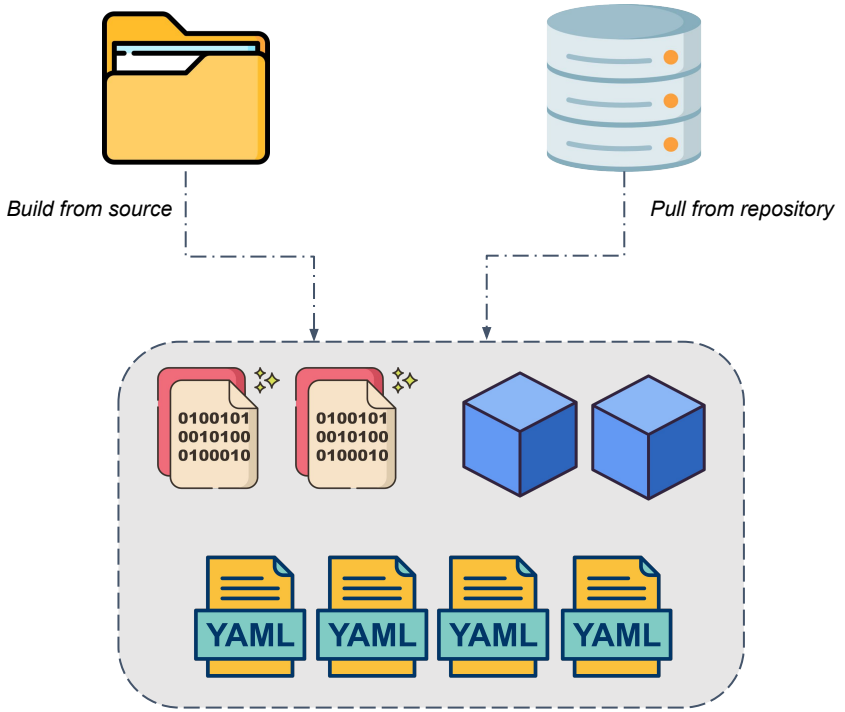


Spin up a development environment



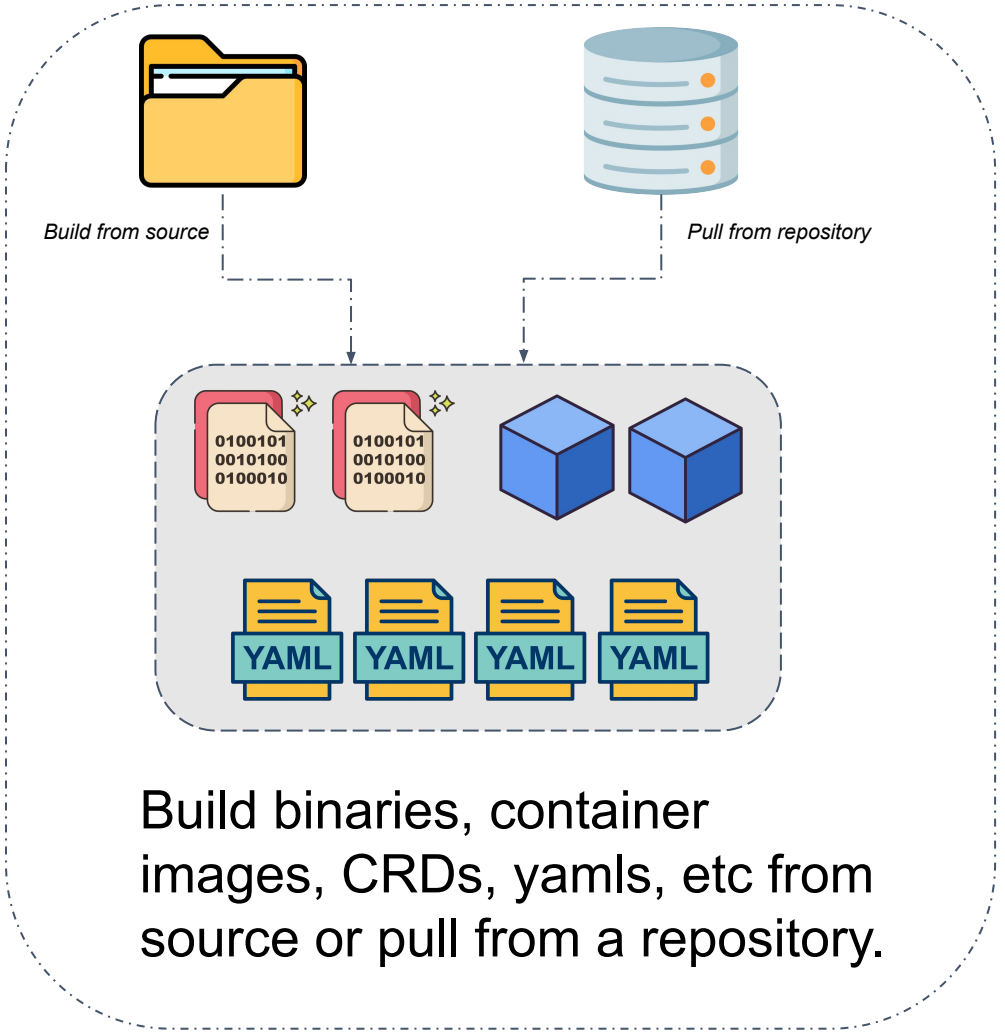
Local Cluster or a Remote Cluster depending on the size of your application.

Spin up a development environment



Build binaries, container images, CRDs, yamls, etc from source or pull from a repository.

Spin up a development environment



Build binaries, container images, CRDs, yamls, etc from source or pull from a repository.

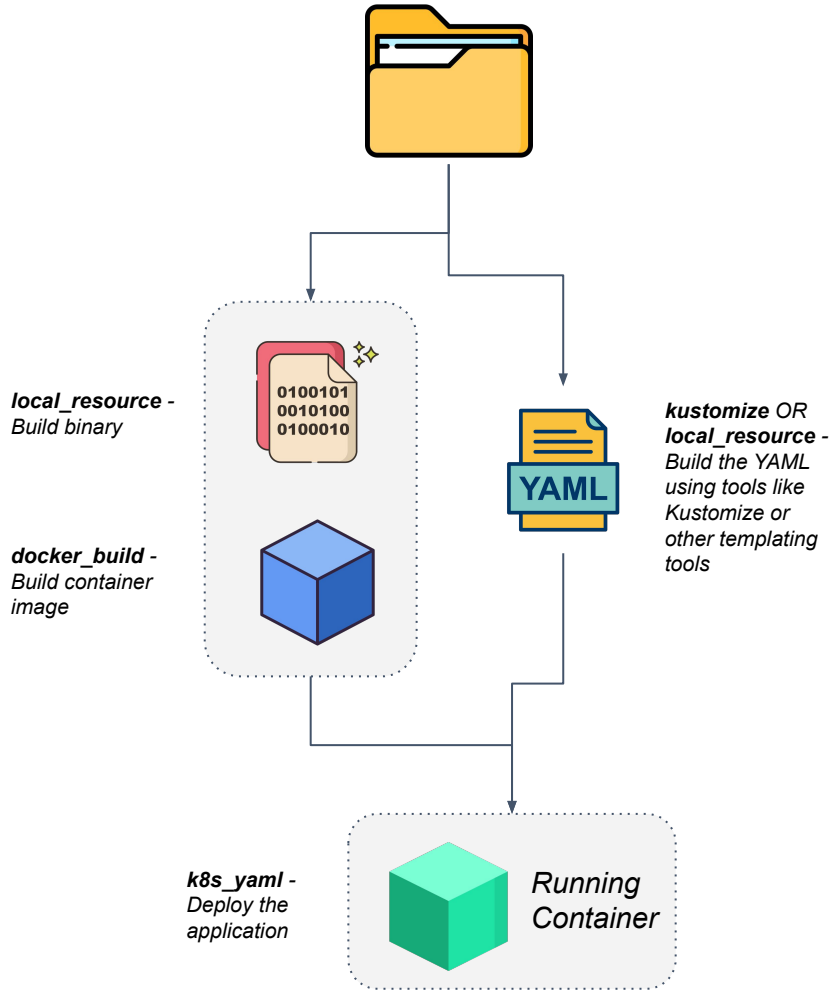
Should be as simple as running ***a single command.***

Spin up a development environment

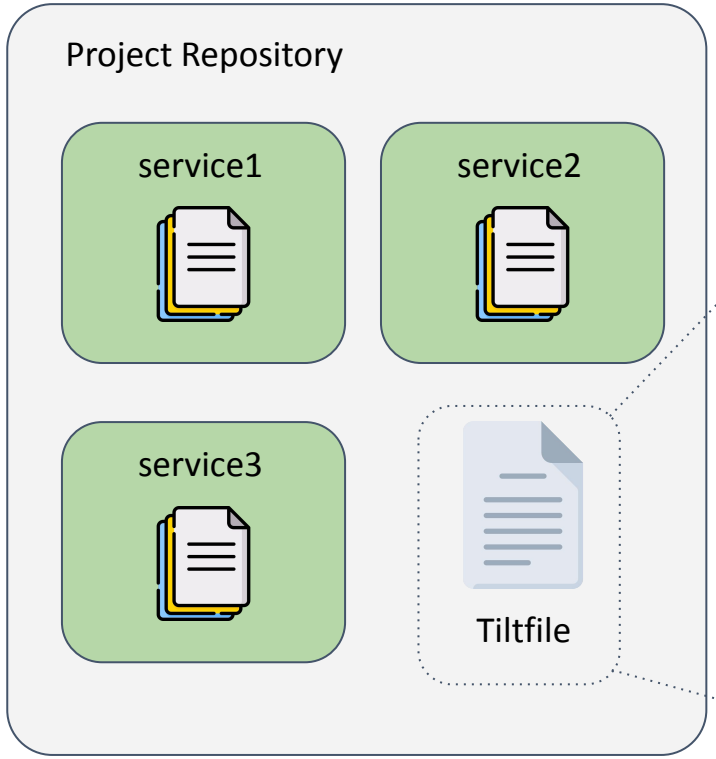
A toolkit for fixing the pains of developing microservices
… and more.

Spin up a development environment

Use a single command `tilt up`
to spin up a development environment.



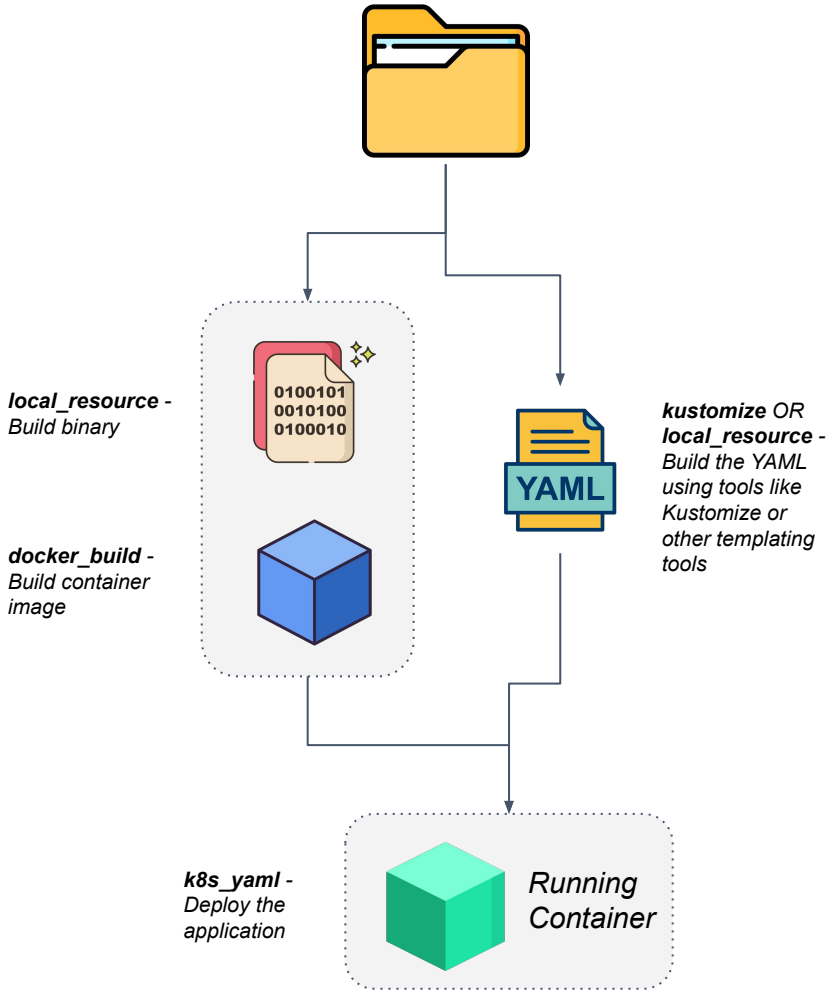
Spin up a development environment




```
// Build the binary
local_resorce(
    cmd = "go build ....",
    labels= ["service1"],
    deps = live_relead_deps,
)

// Build the docker image
docker_build(
    ref = image,
    context = "./service1"
    live_update = [...],
)

// Deploy
k8s_yaml("./yamls/service1.yaml")
```



Tiltfile:

- Written in Starlark - a dialect of Python
- Full power of conditions, loops, functions, etc

What makes a good development workflow?

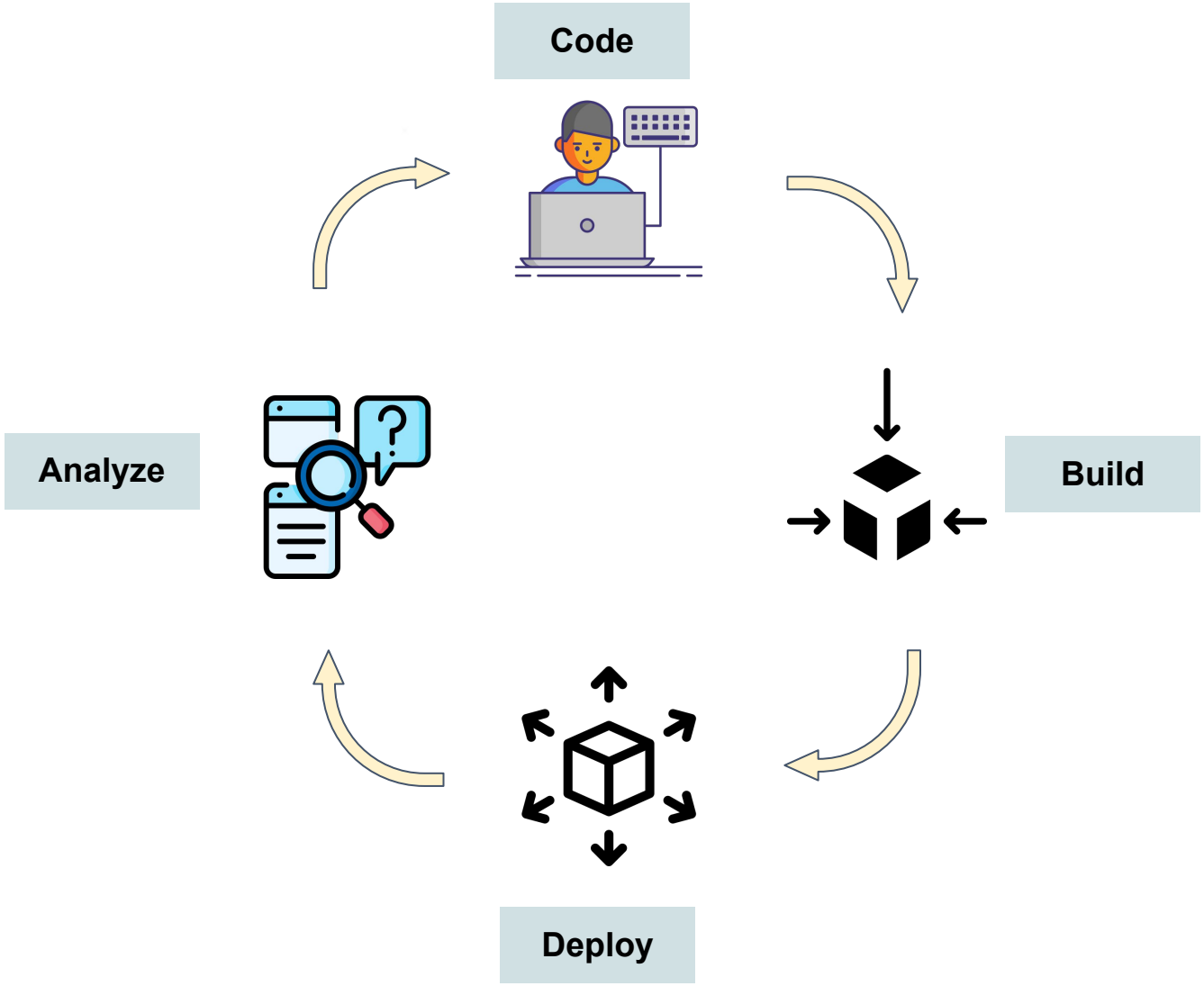
Easy spin up

Quick feedback loop

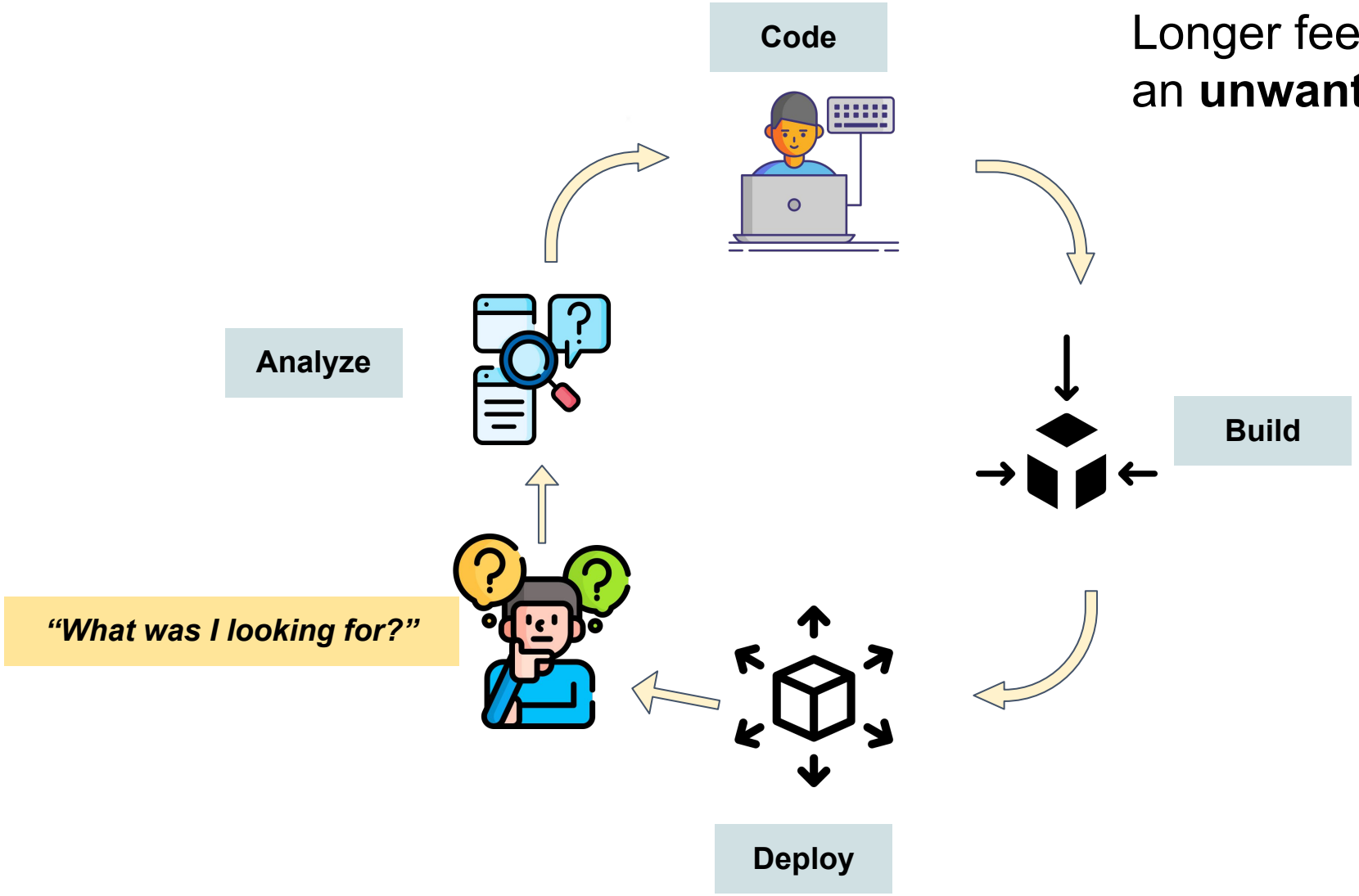
Debuggable

Visibility

Feedback loop



Feedback loop

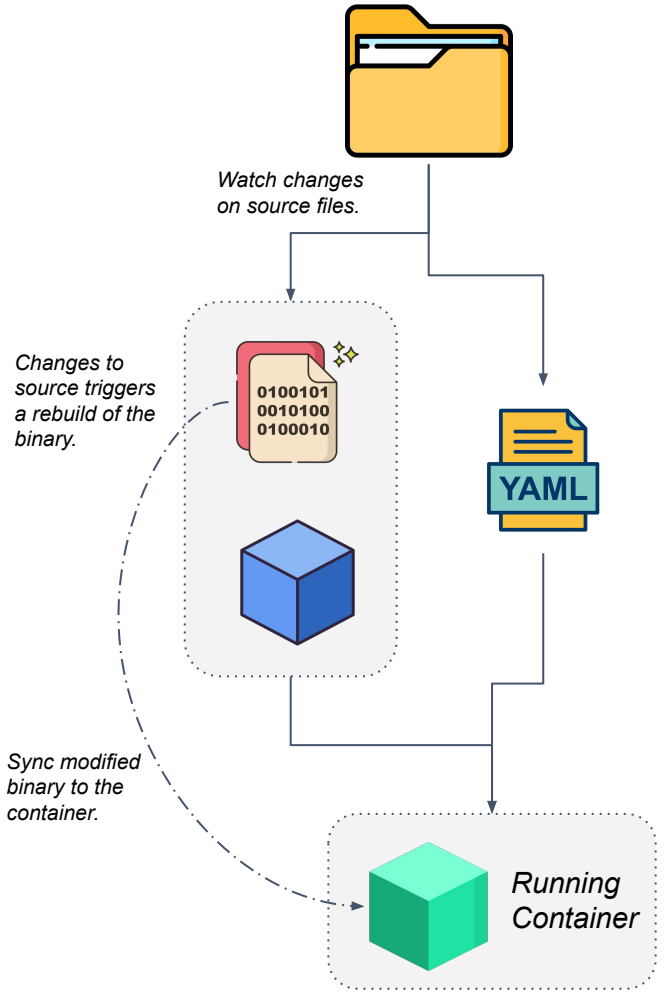


Longer feedback loops add an **unwanted step**.

Feedback loop

Use **Live Update** to hot-reload the application.

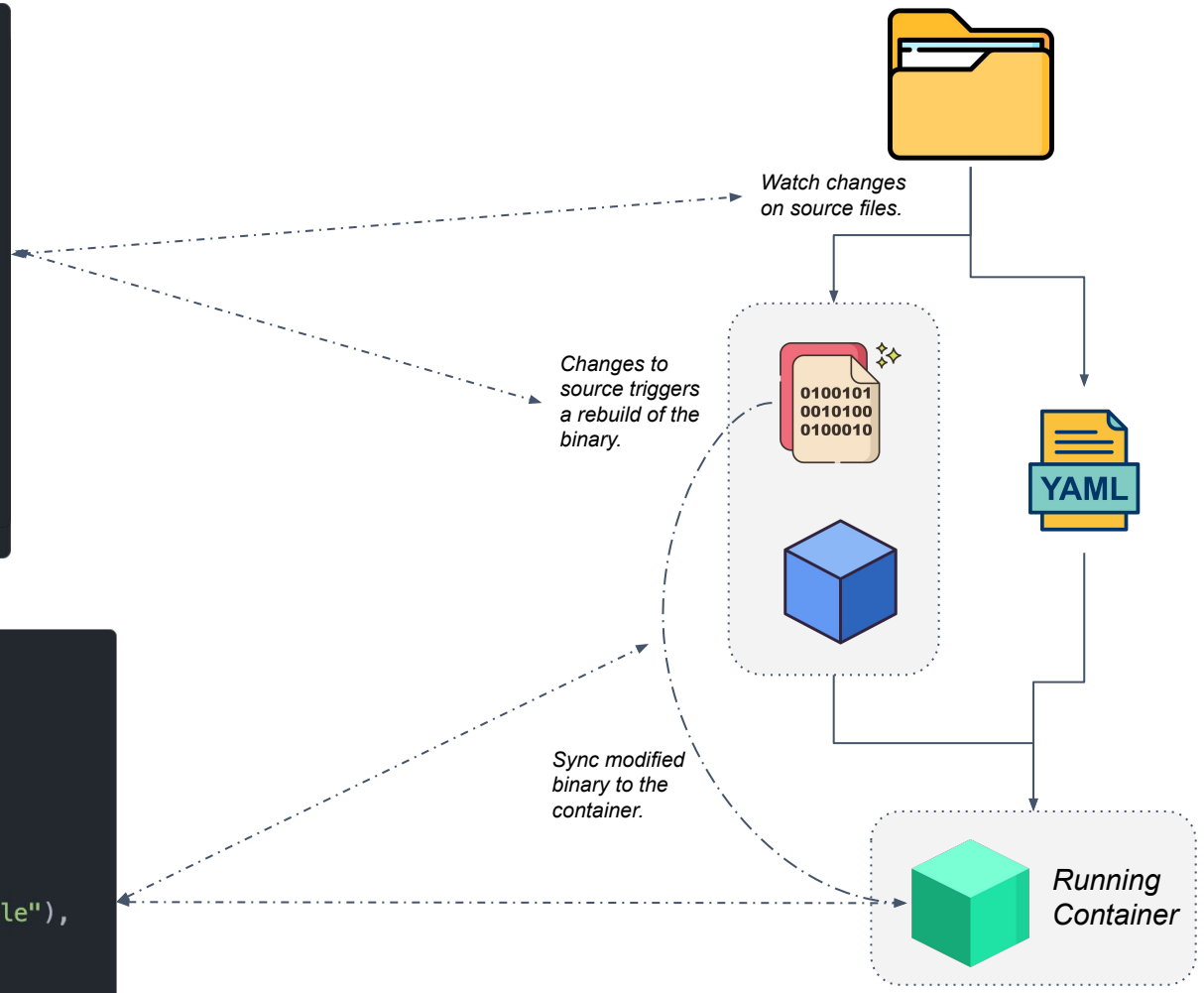
Cut down the feedback loop and see changes as close to real-time as possible.



Feedback loop


```
local_resorce(
    cmd = "go build ....",
    labels= ["service1"],
    deps = [
        "./service1/main.go",
        "./service1/pkg1",
        ...
    ],
)
```

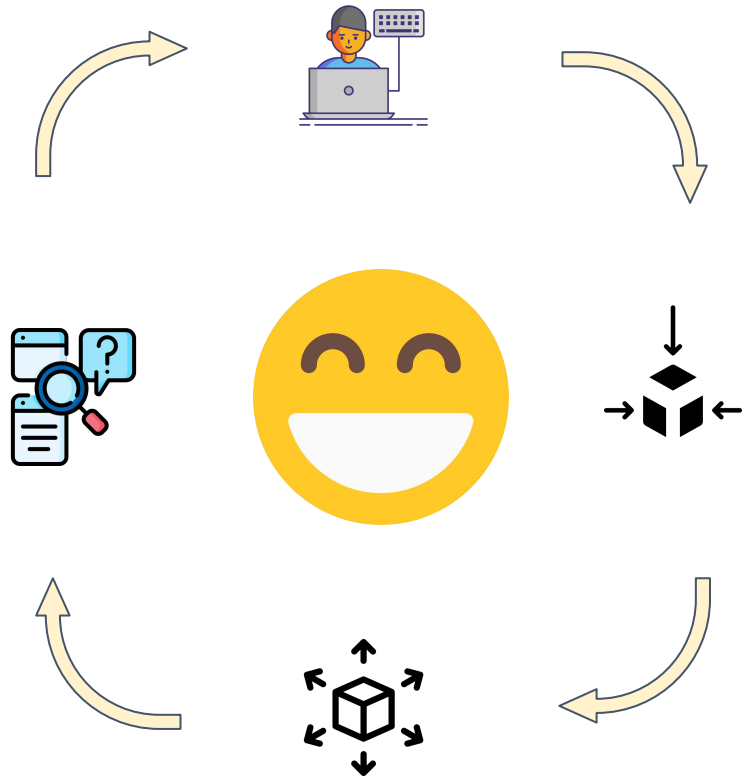
```
docker_build(
    ref = image,
    context = "./service1"
    live_update = [
        // Sync file to conatiner
        sync("./service1/executable", "/executable"),
        // Run after files are synced.
        run("sh /restart.sh"),
    ],
)
```



Feedback loop

No more:

- Rebuild container image `docker build`
- Publish container image `docker push`
- Update image tag in YAML `kubectl apply -f …`
- Wait for pods to rollouts



What makes a good development workflow?

Easy spin up

Quick feedback loop

Debuggable

Visibility

Debugging

*The first place most start debugging is by **adding logs***

*Quickly becomes **overwhelming** and **hard to debug***

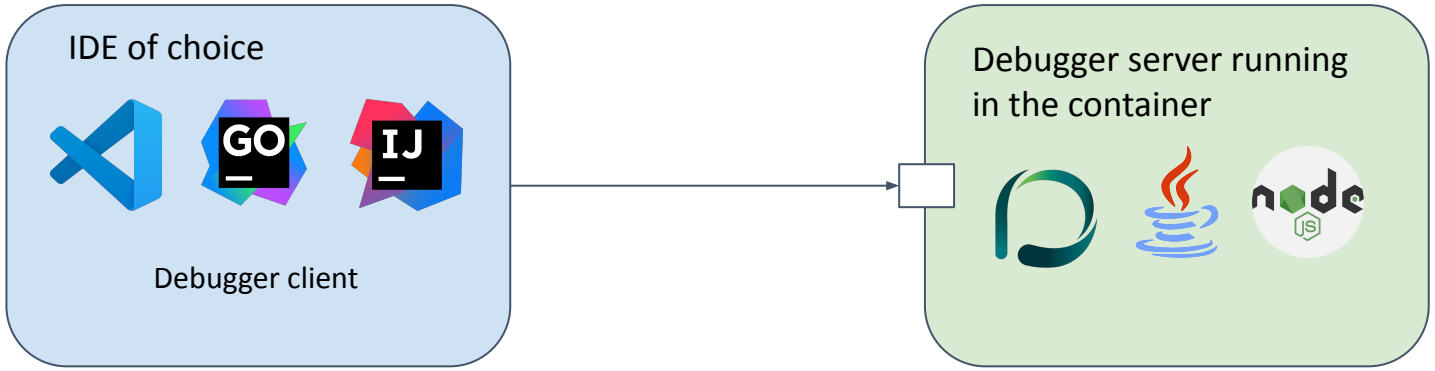
*Using a **debugger** to step through code is the better option*

… but can get tricky when working with containers.

Debugging

Remote Debugging

- Run a debug server remotely
- Use your local client (IDE) to connect to the remote debug server



Debugging

- Build a debugging compatible binary
- Modify the Deployment
- Port-forward to the debug server port
- Connect to the remote server from the debug client

Debugging

- **Build a debugging compatible binary**
  - Disable compile time optimizations
  - Retain the debugging data (DWARF)
- Modify the Deployment
- Port-forward to the debug server port
- Connect to the remote server from the debug client

Debugging

- Build a debugging compatible binary
- **Modify the Deployment**
- Port-forward to the debug server port
- Connect to the remote server from the debug client

- Change the deployment container's command to start the debugging server

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: service1
spec:
  ...
      containers:
      - name: service1
        command:
        - sh
        - /start.sh
        - /dlv
        - --accept-multiclient
        - --api-version=2
        - --headless=true
        - exec
        # Debugger server port
        - --listen=:30000
        - --
        - /executable
        args: ...
        image: ...
```

- Disable liveness and readiness probes

Debugging

- Build a debugging compatible binary
- Modify the Deployment
- **Port-forward to the debug server port**
- Connect to the remote server from the debug client

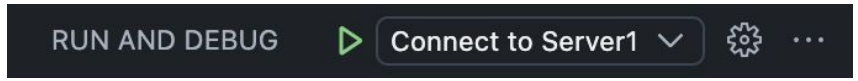
- Setup port forwarding from a free local port to the debug port of the container

```
k8s_resource(
    workload = "service1",
    labels = ["service1"],
    // Port foward to the debug on the container
    port_forwards = [
        port_forward("30000", "30000"),
    ],
)
```

Debugging

- Build a debugging compatible binary
- Modify the Deployment
- Port-forward to the debug server port
- **Connect to the remote server from the debug client**

- Connect to the remote debugger using the debug client (built-into most IDEs)

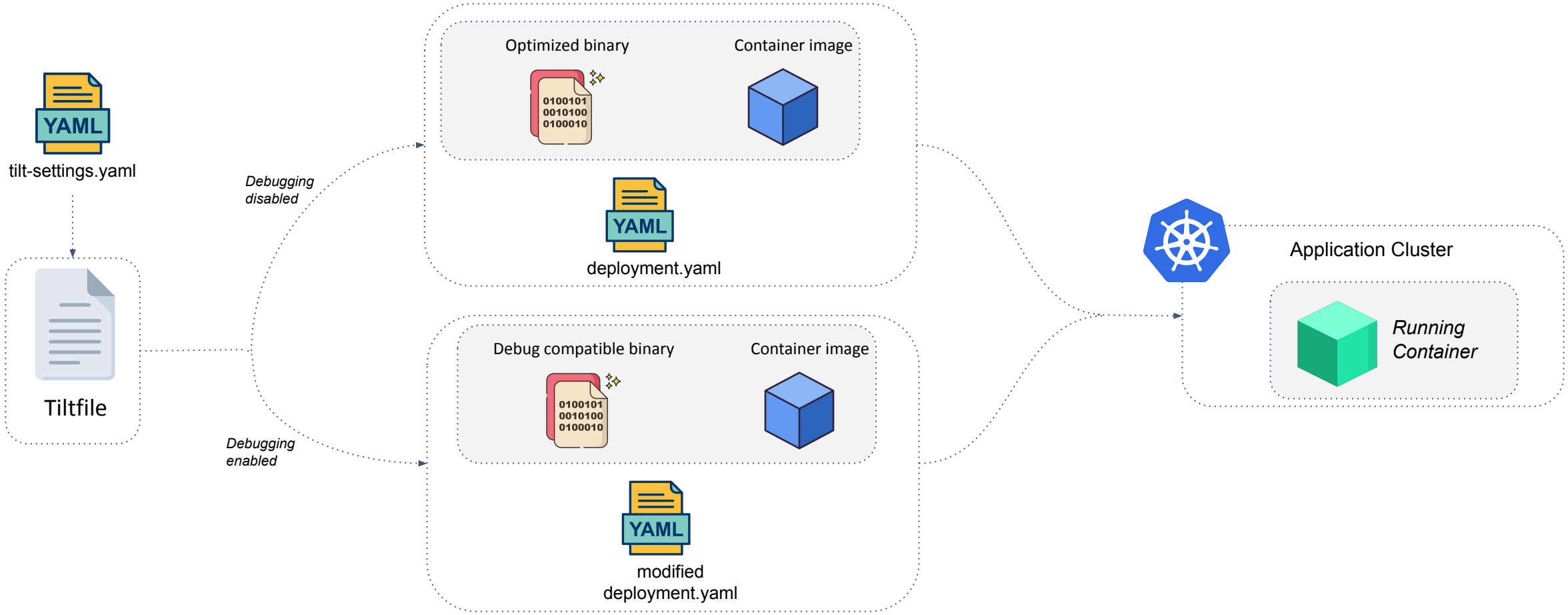



```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Connect to Server1",
            "type": "go",
            "request": "attach",
            "mode": "remote",
            // Port setup for port-forwarding
            "remotePath": "",
            "port": 30000,
            "host": "127.0.0.1"
        }
    ]
}
```

Debugging

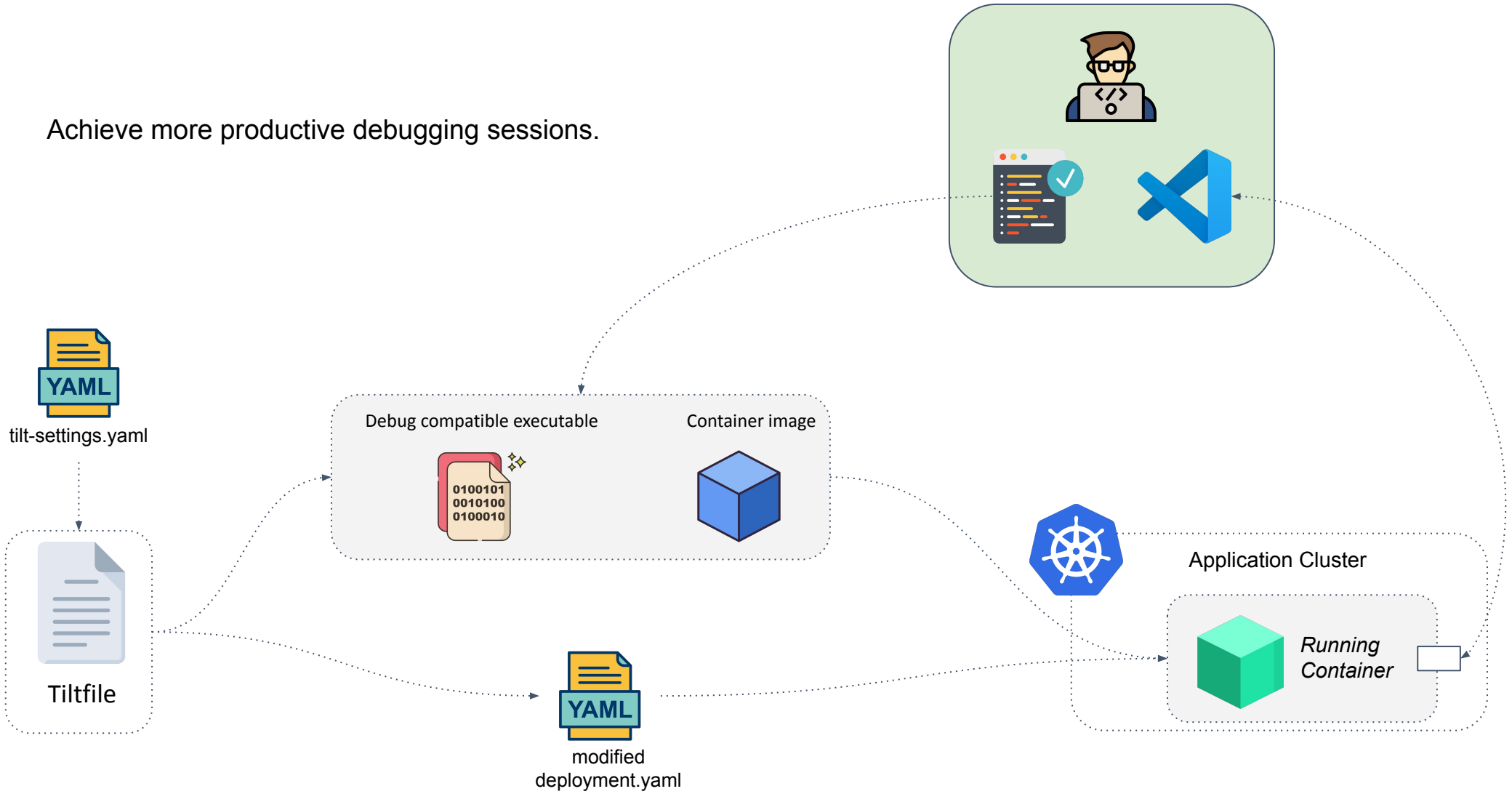
No need to run the debugger all the time. Enable it on demand.

Have a `tilt-settings.yaml` to configure debugging parameters.



Debugging

Achieve more productive debugging sessions.



What makes a good development workflow?

Easy spin up

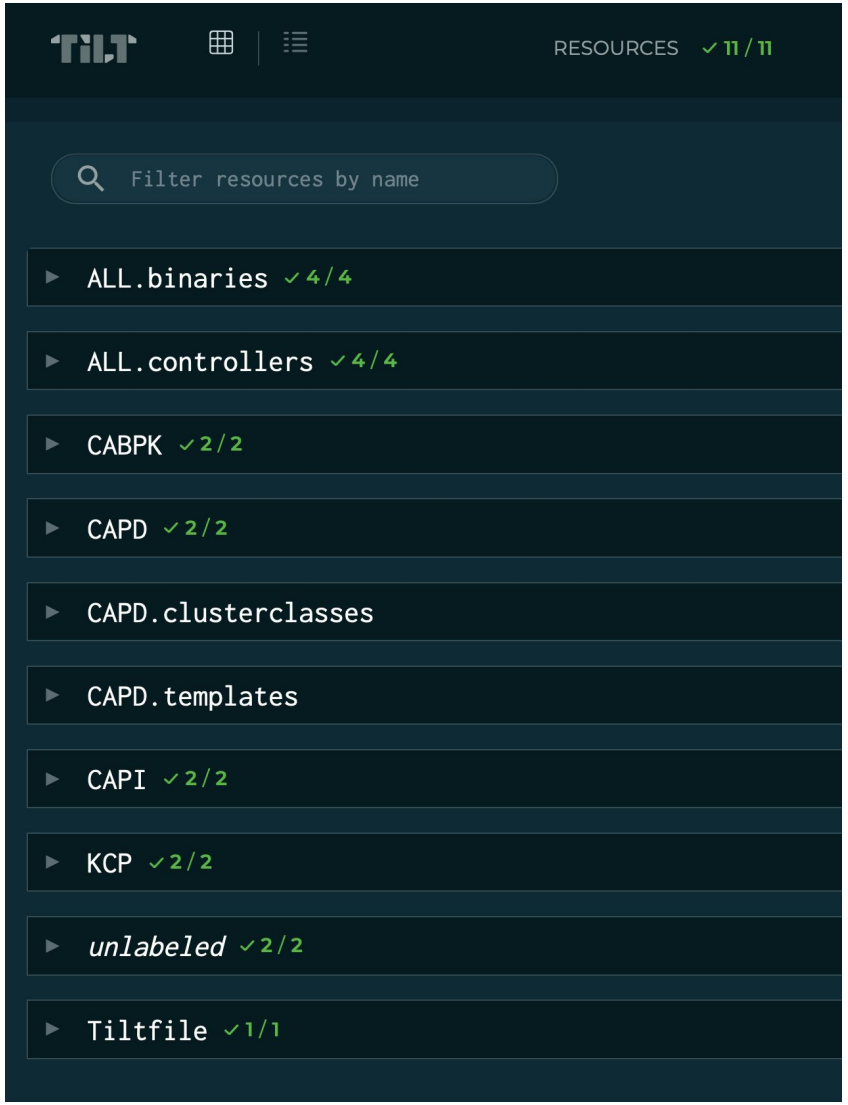
Quick feedback loop

Debuggable

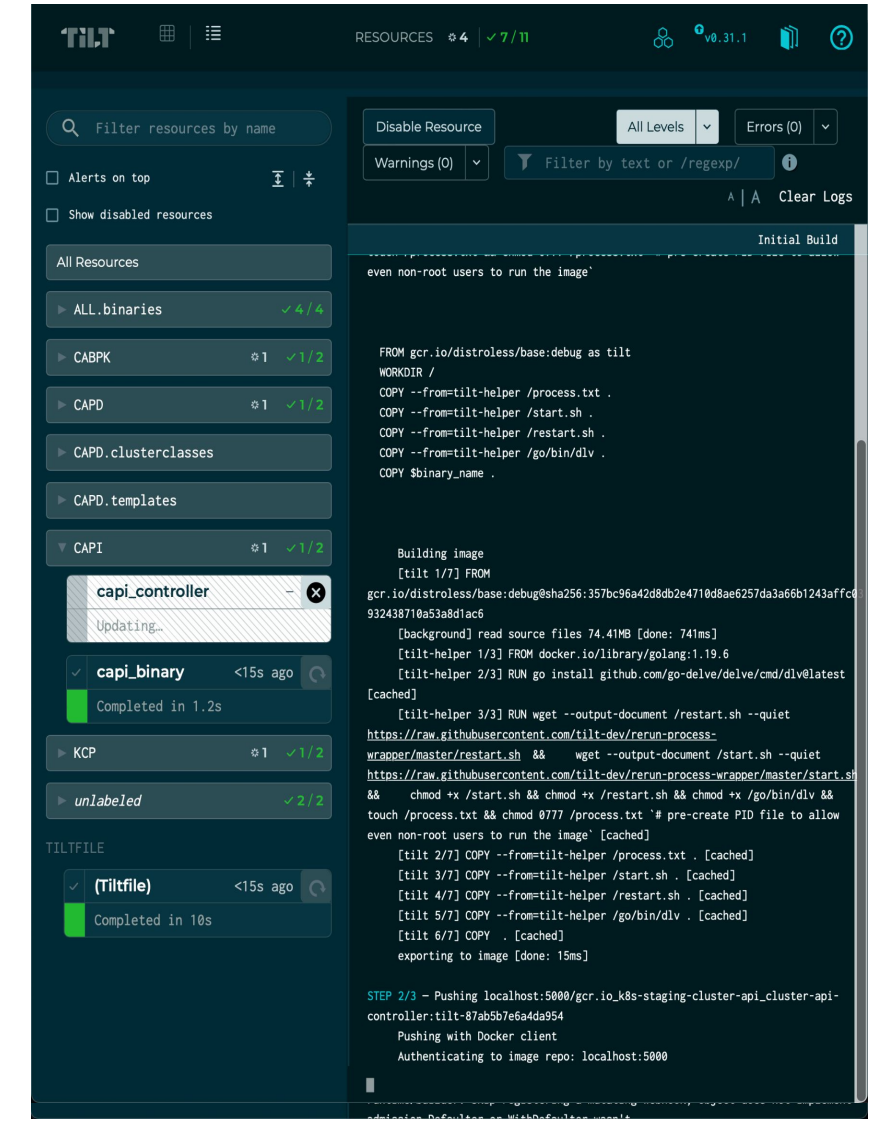
Visibility

Visibility

Showcases overall status of the application



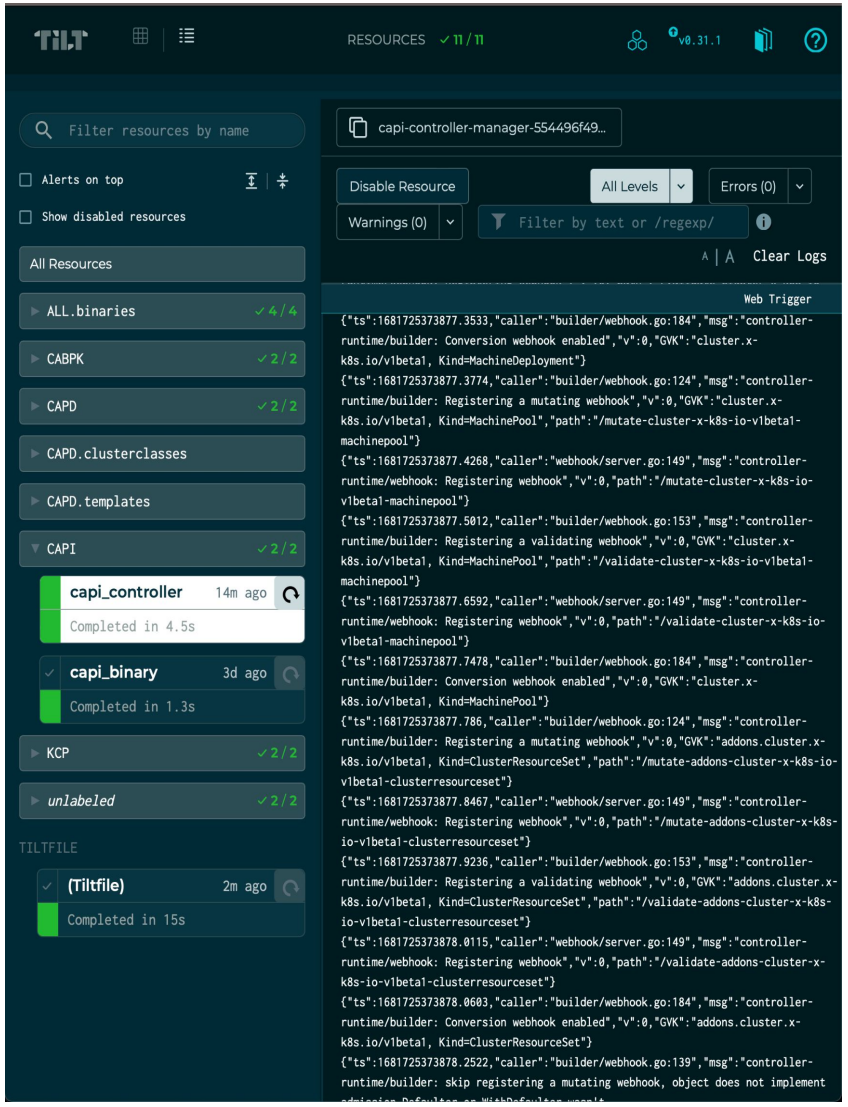
Visibility



Showcases overall status of the application

Follow live updates to the components

Visibility

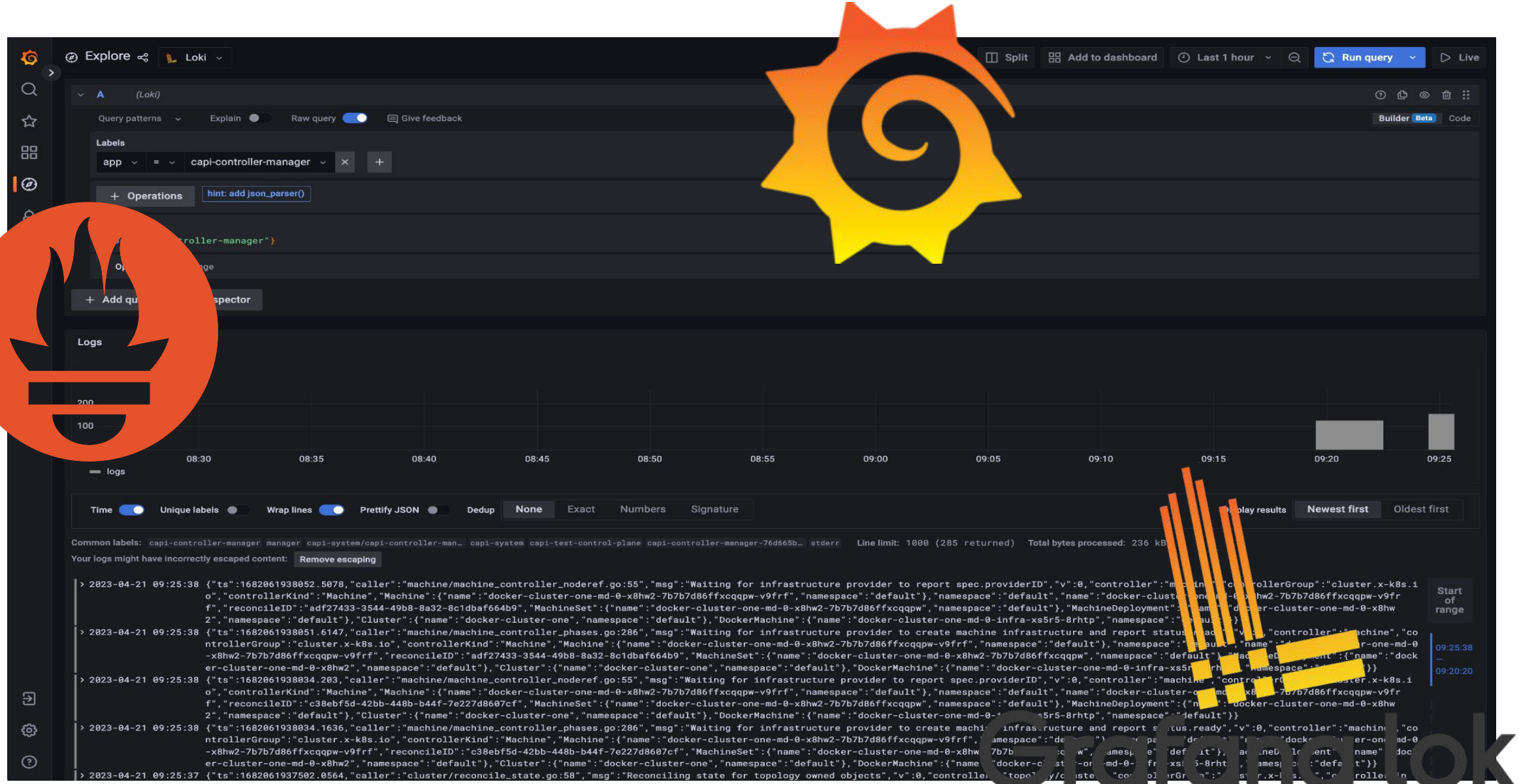


Showcases overall status of the application

Follow live updates to the components

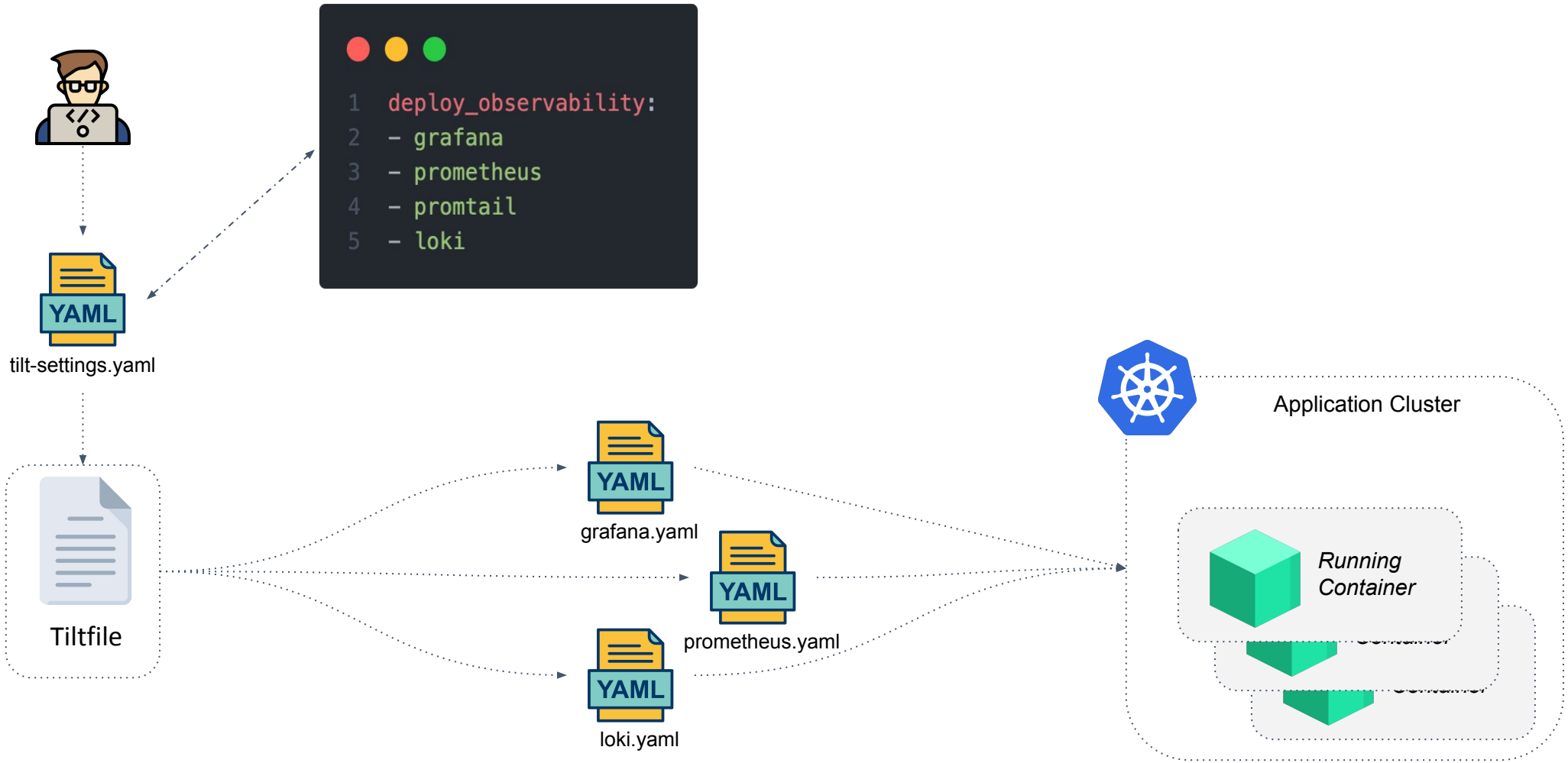
Provides detailed information for each component

Visibility


Explore
Loki
Split
Add to dashboard
Last 1 hour
Run query
Live
A (Loki)
Query patterns
Explain
Raw query
Give feedback
Builder Beta
Code
Labels
app
capi-controller-manager
Operations
hint: add json_parser()
Logs
200
100
08:30
08:35
08:40
08:45
08:50
08:55
09:00
09:05
09:10
09:15
09:20
09:25
logs
Time
Unique labels
Wrap lines
Prettify JSON
Dedup
None
Exact
Numbers
Signature
Newest first
Oldest first
Common labels:
Your logs might have incorrectly escaped content:
Remove escaping
Start of range
Grafana loki

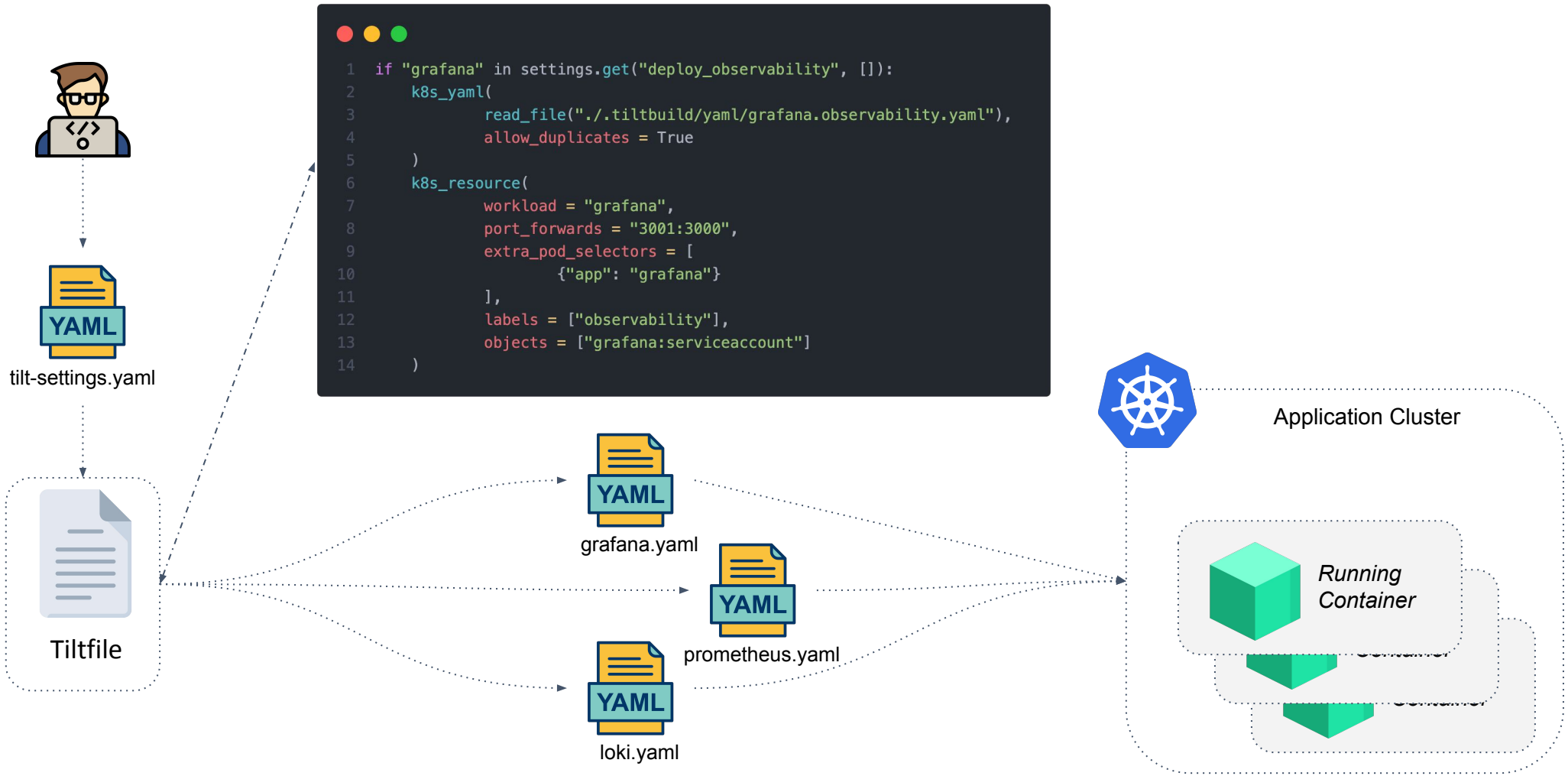
Visibility

Enable the observability tools on demand



Visibility

Enable the observability tools on demand



What makes a good development workflow?

Finally…

What makes a good development workflow?

Easy spin up

`tilt up`

Quick feedback loop

Debuggable

Visibility

What makes a good development workflow?

Easy spin up → `tilt up`

Quick feedback loop → Live Update

Debuggable

Visibility

What makes a good development workflow?

Easy spin up → `tilt up`

Quick feedback loop → Live Update

Debuggable → Remote Debugger

Visibility

What makes a good development workflow?

Easy spin up → `tilt up`

Quick feedback loop → Live Update

Debuggable → Remote Debugger

Visibility → UI & Observability tools


KubeCon
CloudNativeCon
Europe 2023
Thank you!