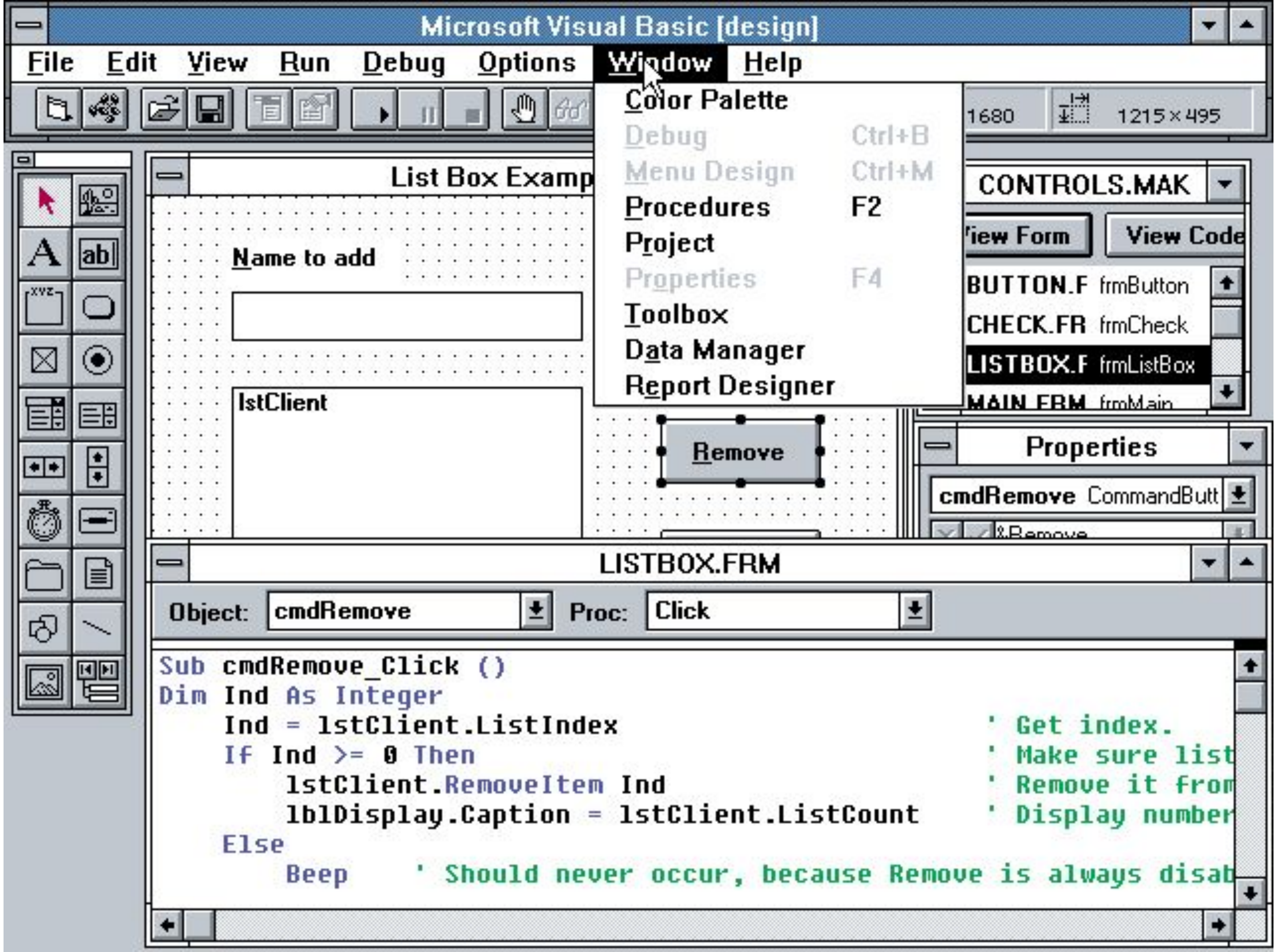
Successful Kubernetes Development Workflows

Ellen Körbes

TILT



```
Sub cmdRemove_Click ()
Dim Ind As Integer
    Ind = lstClient.ListIndex                         ' Get index.
    If Ind >= 0 Then                                  ' Make sure list
        lstClient.RemoveItem Ind                      ' Remove it from
        lblDisplay.Caption = lstClient.ListCount      ' Display number
    Else
        Beep    ' Should never occur, because Remove is always disab
```

Syntax Highlight

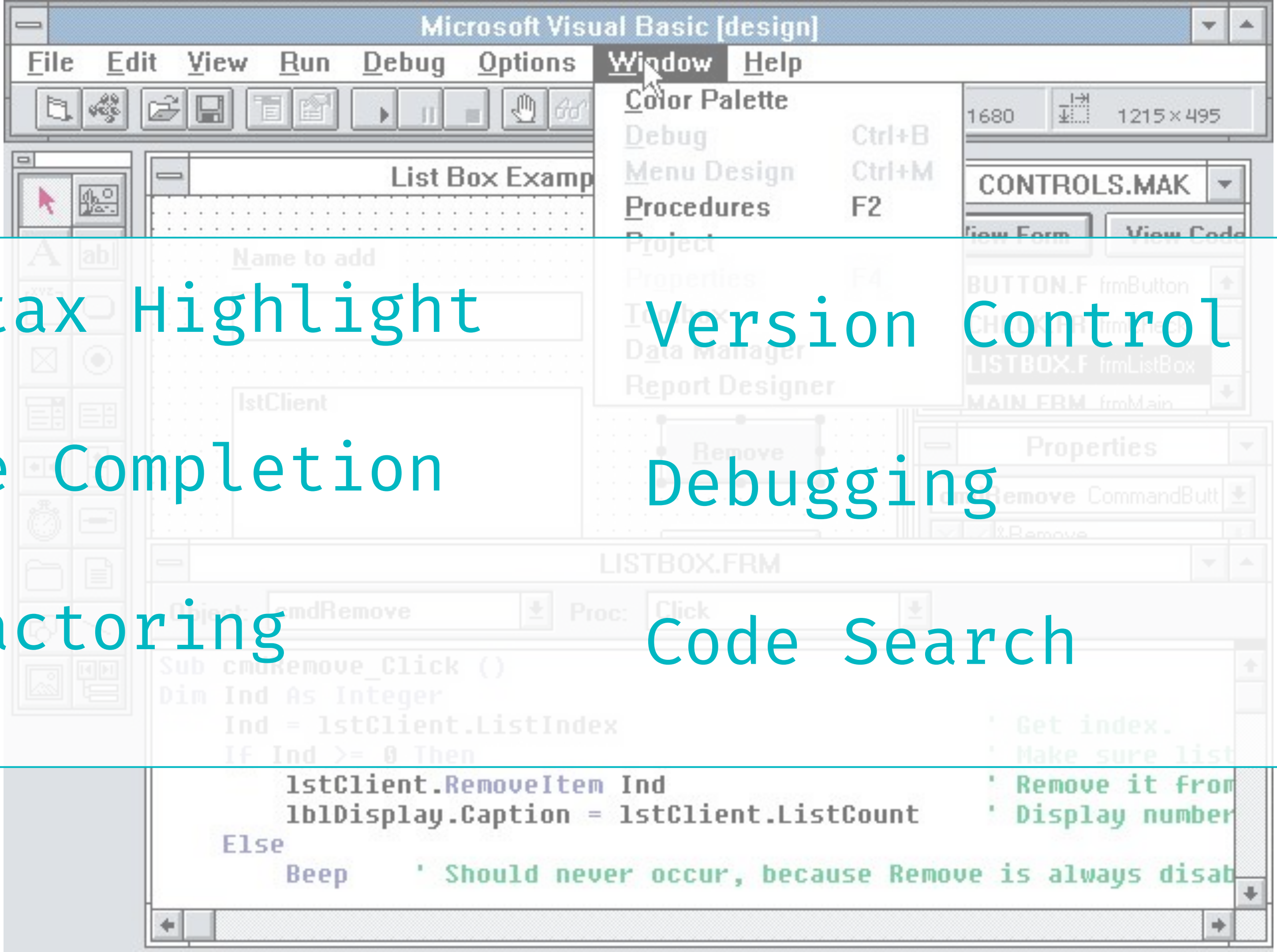
Version Control

Code Completion

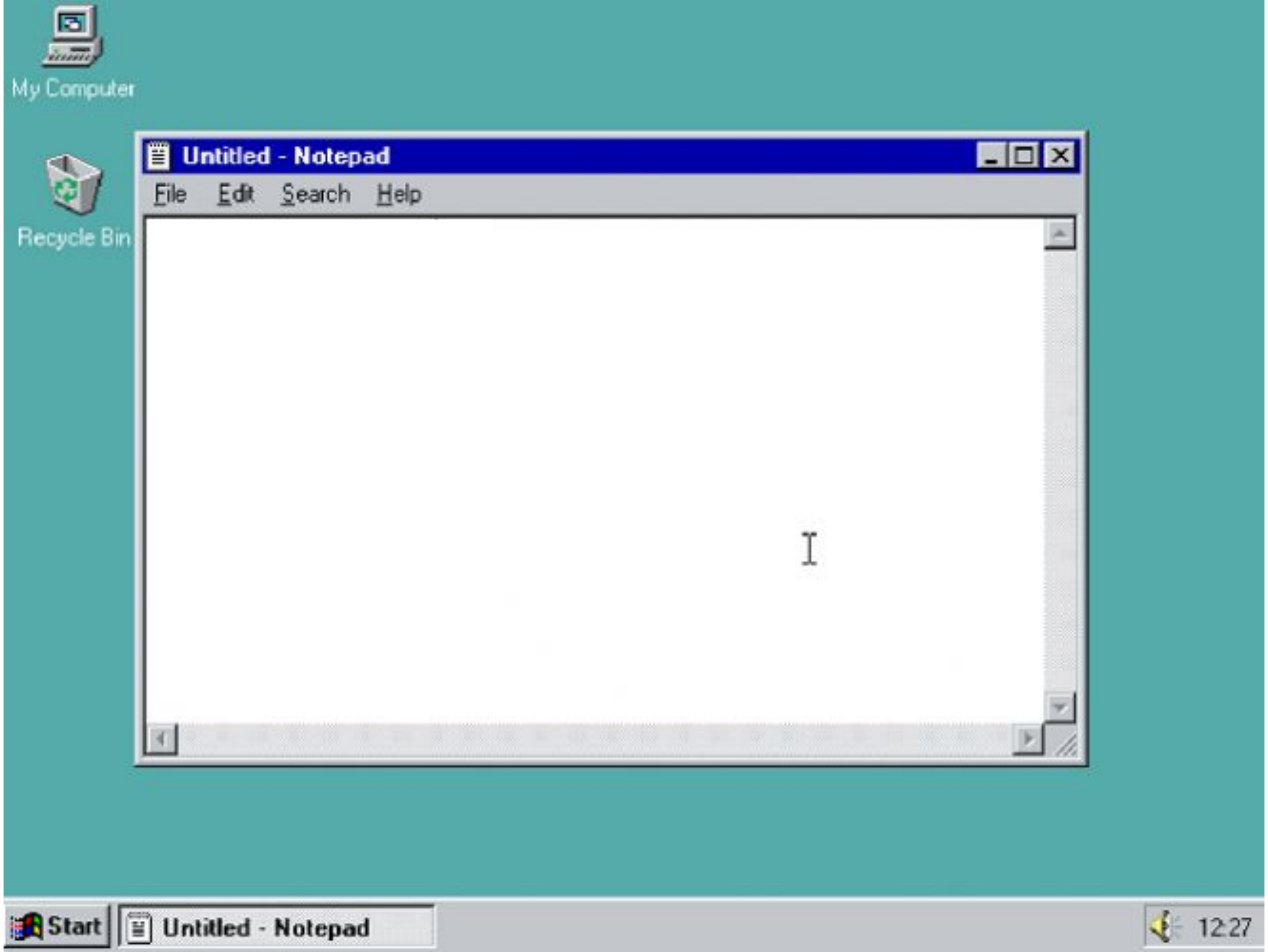
Debugging

Refactoring

Code Search



```
Sub cmdRemove_Click ()
Dim Ind As Integer
    Ind = lstClient.ListIndex                        ' Get index.
    If Ind >= 0 Then                                 ' Make sure list
        lstClient.RemoveItem Ind                     ' Remove it from
        lblDisplay.Caption = lstClient.ListCount     ' Display number
    Else
        Beep    ' Should never occur, because Remove is always disab
```


My Computer
Recycle Bin
Untitled - Notepad
File
Edit
Search
Help
Start
Untitled - Notepad
12:27

The Problem:

Containers and Kubernetes are incredible!

```
kubectl run --restart=Always    # creates deployment

kubectl run --restart=Never     # creates pod

kubectl run --restart=OnFailure # creates job
```

The Problem:

Containers and Kubernetes are incredible!

…except for the **development workflow**.

$ whoami

- **Ellen Körbes**
  - CNCF Ambassador
  - Google Developer Expert for Go
- Focused on the developer experience side of Kubernetes
- Frequent speaker... everywhere 😅

Head of Product

- l@tilt.dev
- @ellenkorbes
- they/she
- #tilt@slack.k8s.io

The Problem Set

The Problem Set

Development Clusters

Managing Configuration Files

Feedback Loop Automation

Problem Solving

Custom Workflows

The Protagonists

The Protagonists

DATADOG

Datadog

Cloud monitoring
SaaS provider.

Engineering team:
~**800** devs.

The Protagonists

DATADOG

Datadog

Cloud monitoring SaaS provider.

Engineering team: ~**800** devs.

unu

unu

Electric scooters manufacturer in Berlin.

Development team has **~25** engineers.

The Protagonists

DATADOG

Datadog

Cloud monitoring SaaS provider.

Engineering team: ~**800** devs.

UNU

unu

Electric scooters manufacturer in Berlin.

Development team has **~25** engineers.

MINDSPACE.

Mindspace

Creative learning & gamification agency.

Very **tiny**!

Four engineers!

The Protagonists

Datadog

Cloud monitoring SaaS provider.

Engineering team: ~**800** devs.

unu

Electric scooters manufacturer in Berlin.

Development team has **~25** engineers.

Mindspace

Creative learning & gamification agency.

Very **tiny**!

Four engineers!

Cluster API

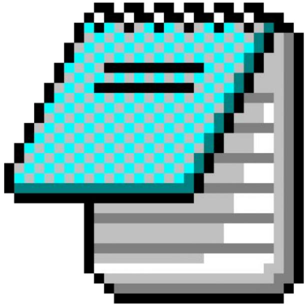
Use a cluster to create, configure, and manage other clusters.

230+ contributors.

Very. Weird. Workflow!

The Problem

Development Clusters

no dev cluster == 

Concerns

Local Cluster:

- Can the whole app fit a laptop’s **RAM**?
- Which **type**? There’s Minikube, kind, Microk8s, etc.
- Double-click setup
- Feedback bottleneck:
  - 1. Compute
  - 2. Network

Concerns

Remote Cluster:

- Cash money dollars.
- Requires **more infra & setup** out of the box.
- No double-click setup!
- Feedback bottleneck:
  - 1. Network
  - 2. Compute

Datadog

- **300+ services**—won't fit a laptop
- Self-managed cluster on **public cloud**
- **Separate namespaces** per team or per developer
- Wrapper tools for provisioning
- At first devs use staging services, which are cloned when working on them
- Option to add debugging tools

Mindspace

- Microk8s on Linux
- Mostly **Docker for Mac**
- **Docker Compose** for unit/integration tests… because Kubernetes in CI.
- Local clusters mirror prod, except for e.g. Mongo, replication

KUBERNETES ON THE LAPTOP IS FINE!!!!!111

...when you do it right.

unu

- **Docker for Mac** (Troublesome!)
- Run everything **locally**
- Hitting **limits**!
- Solution: **optional services**

Cluster API

- Concept: **management clusters**
- Use **kind** as the local dev cluster
- Why? Quick & **easy to tear** down
- Specific development **on every cloud**

Takeaway

Small companies? **Local** cluster.*

Big companies? **Remote** cluster.

Local clusters are easier to start, and companies migrate to remote once things don't fit a laptop anymore.

Takeaway

Small companies? **Local** cluster.*

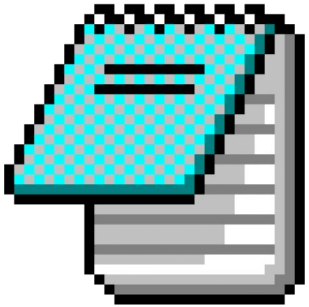
Big companies? **Remote** cluster.

Local clusters are easier to start, and companies migrate to remote once things don't fit a laptop anymore.

* Don’t know which? Check out dex.dev!

The Problem

Managing Configuration Files

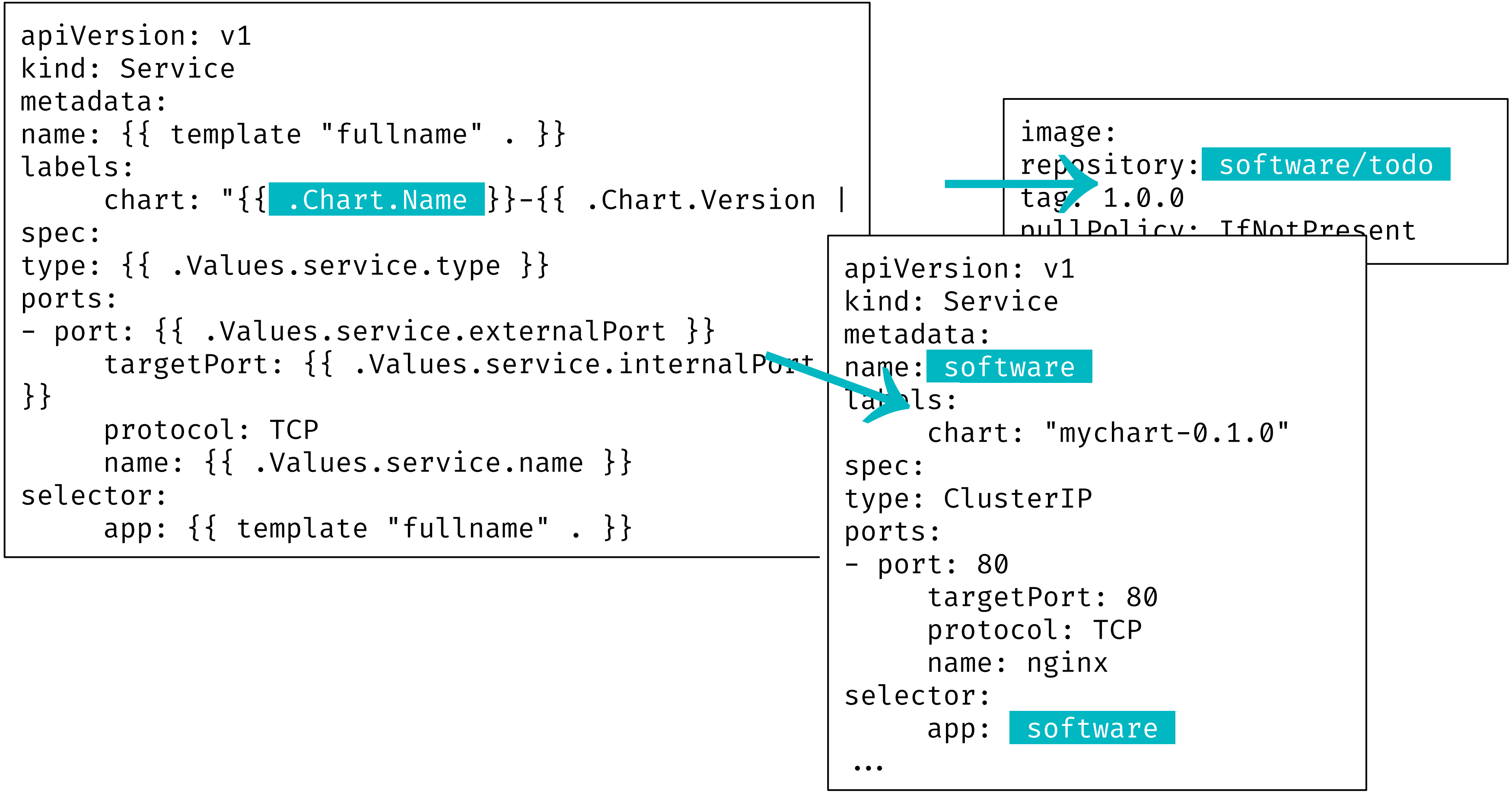
manually editing yaml == 

Why?

Consider a simple Kubernetes app. YAML files:

- Deployment
- Service
- PersistentVolume
- StatefulSet
- Ingress

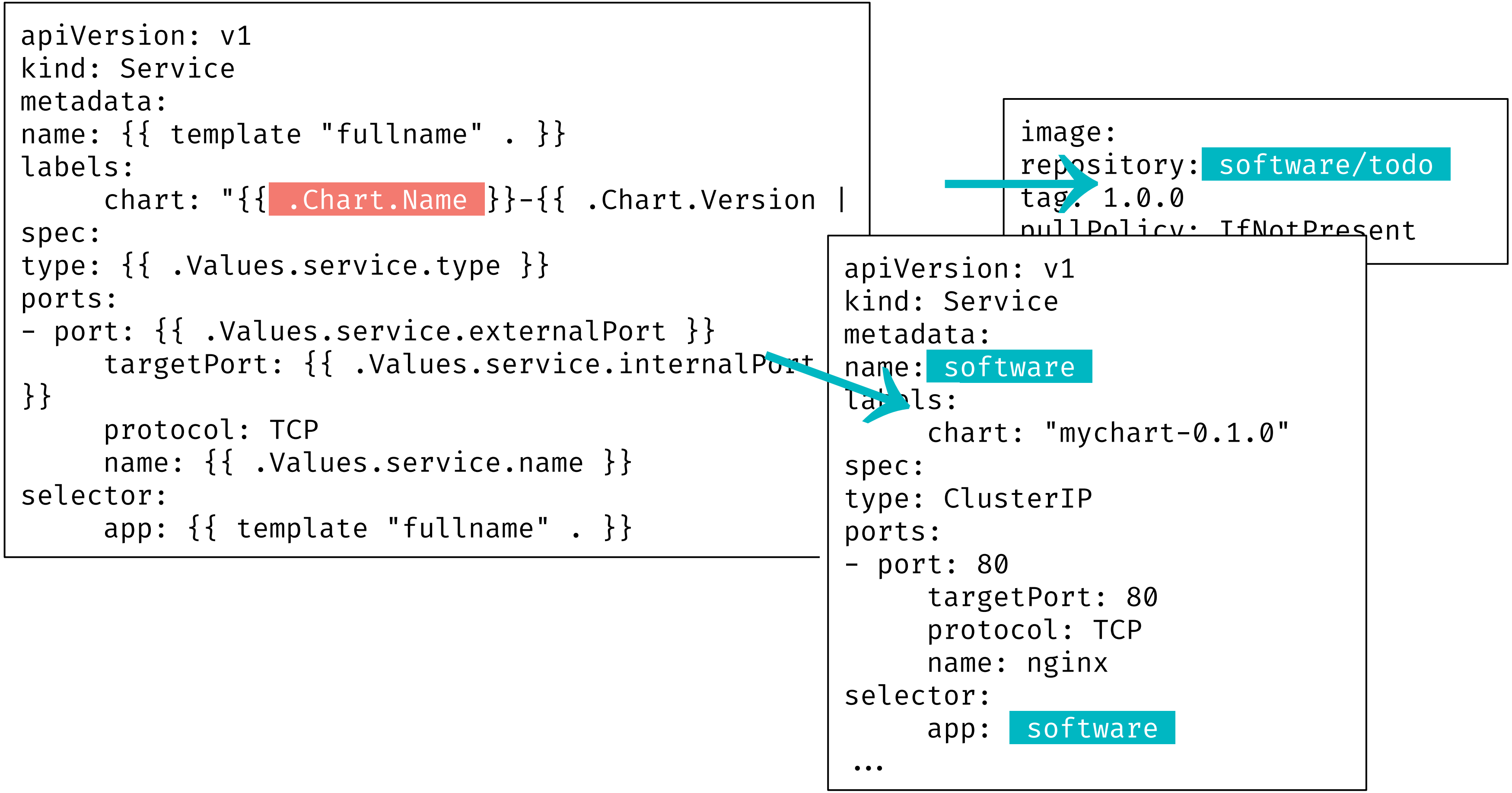
It goes on...



```
apiVersion: v1
kind: Service
metadata:
name: {{ template "fullname" . }}
labels:
     chart: "{{ .Chart.Name }}-{{ .Chart.Version |
spec:
type: {{ .Values.service.type }}
ports:
- port: {{ .Values.service.externalPort }}
     targetPort: {{ .Values.service.internalPort
}}
     protocol: TCP
     name: {{ .Values.service.name }}
selector:
     app: {{ template "fullname" . }}
```

```
image:
repository: software/todo
tag: 1.0.0
pullPolicy: IfNotPresent
```

```
apiVersion: v1
kind: Service
metadata:
name: software
labels:
     chart: "mychart-0.1.0"
spec:
type: ClusterIP
ports:
- port: 80
     targetPort: 80
     protocol: TCP
     name: nginx
selector:
     app: software
 ...
```

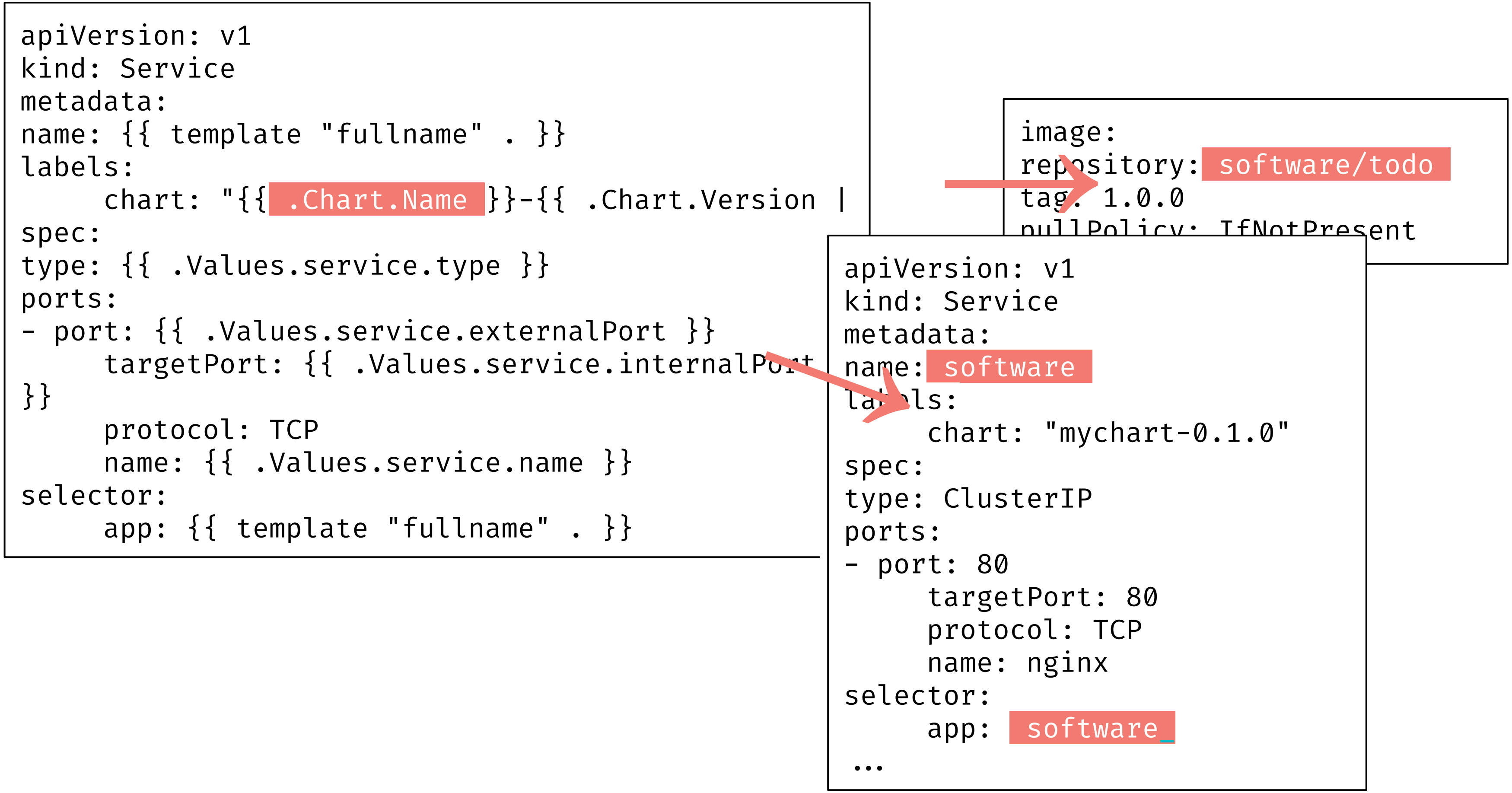


```
apiVersion: v1
kind: Service
metadata:
name: {{ template "fullname" . }}
labels:
     chart: "{{ .Chart.Name }}-{{ .Chart.Version |
spec:
type: {{ .Values.service.type }}
ports:
- port: {{ .Values.service.externalPort }}
     targetPort: {{ .Values.service.internalPort
}}
     protocol: TCP
     name: {{ .Values.service.name }}
selector:
     app: {{ template "fullname" . }}
```

```
image:
repository: software/todo
tag: 1.0.0
pullPolicy: IfNotPresent
```

```
apiVersion: v1
kind: Service
metadata:
name: software
labels:
     chart: "mychart-0.1.0"
spec:
type: ClusterIP
ports:
- port: 80
     targetPort: 80
     protocol: TCP
     name: nginx
selector:
     app: software
 ...
```

@ellenkorbes



```
apiVersion: v1
kind: Service
metadata:
name: {{ template "fullname" . }}
labels:
     chart: "{{ .Chart.Name }}-{{ .Chart.Version |
spec:
type: {{ .Values.service.type }}
ports:
- port: {{ .Values.service.externalPort }}
     targetPort: {{ .Values.service.internalPort
}}
     protocol: TCP
     name: {{ .Values.service.name }}
selector:
     app: {{ template "fullname" . }}
```

```
image:
repository: software/todo
tag: 1.0.0
pullPolicy: IfNotPresent
```

```
apiVersion: v1
kind: Service
metadata:
name: software
labels:
     chart: "mychart-0.1.0"
spec:
type: ClusterIP
ports:
- port: 80
     targetPort: 80
     protocol: TCP
     name: nginx
selector:
     app: software
 ...
```

Datadog

- **Helm** templates
- Different values for different environments (1 node vs. 100)
- **One dev writes the YAML** the first time...
- ...everyone else just `tilt` ups.

Mindspace

- Services follow a **common pattern**
- **Helm templating** creates YAML
- Helm is further **automated with Tilt**

unu

- Services follow a **common pattern**
- **Helm templating** creates YAML
- Helm is further **automated with Tilt**
- Semi-custom Tilt/Bash YAML generator. Multi-layered Helm values file so people can override values per service, env, or locally.

Cluster API

- **Convention** for all provider projects: Provider-specific JSON.
- User-specific Tilt settings on tilt-settings.json **overlays** on top of **defaults**.
- **Kustomize** templating
- For development everything is extremely **uniform**.

Takeaways

Everyone uses a templating solution.*

Big companies sometimes roll their own.

Almost everyone uses **Helm** templates.

Takeaways

Everyone uses a templating solution.*

Big companies sometimes roll their own.

Almost everyone uses **Helm** templates.

* Don’t know which to choose? Check out dex.dev!

The Problem

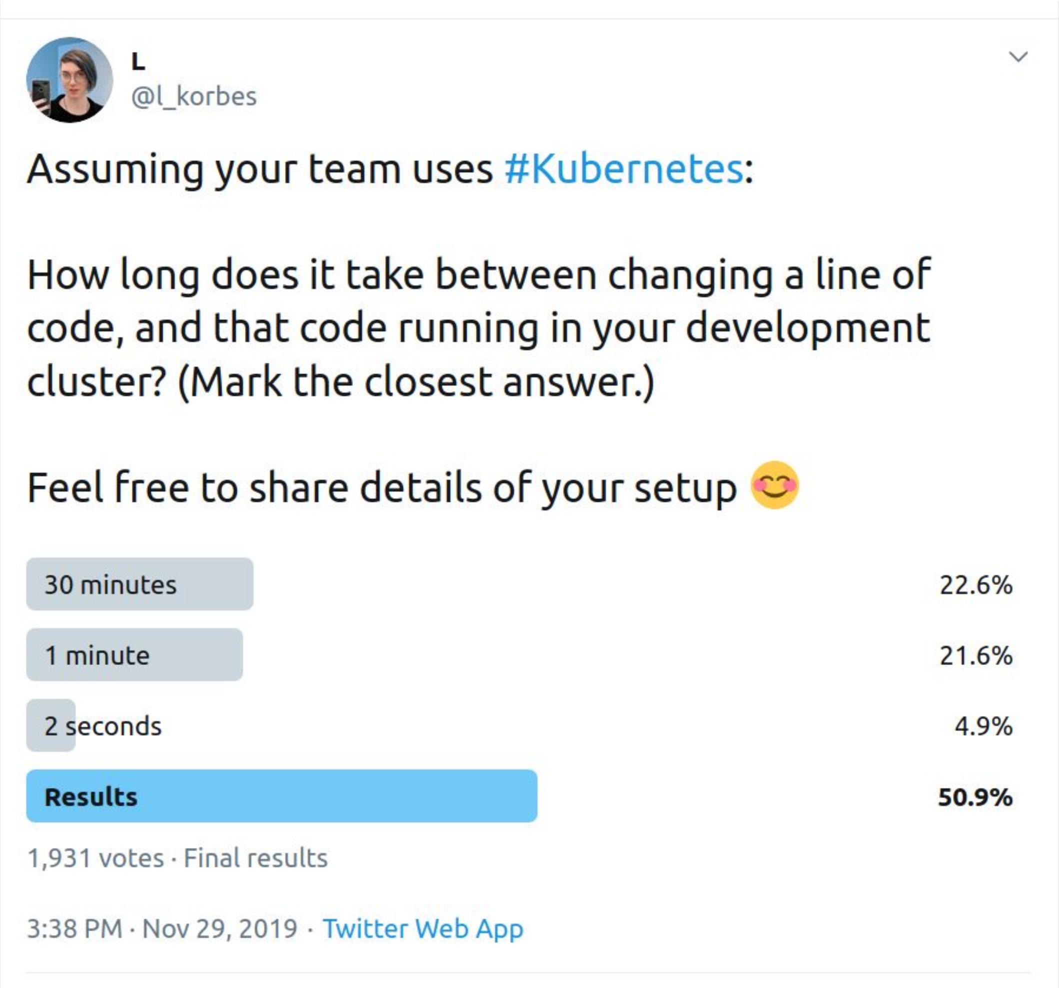
Feedback Loop Automation

What?

Roughly, we want the following operations:

- docker build
- docker push
- kubectl apply

...to be done automatically.



L
@l_korbes

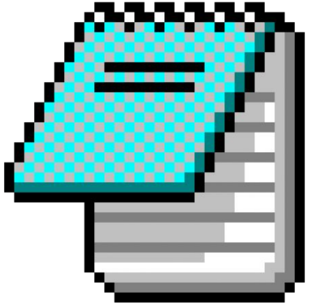
Assuming your team uses #Kubernetes:

How long does it take between changing a line of code, and that code running in your development cluster? (Mark the closest answer.)

Feel free to share details of your setup 😊

30 minutes	22.6%
1 minute	21.6%
2 seconds	4.9%
Results	**50.9%**

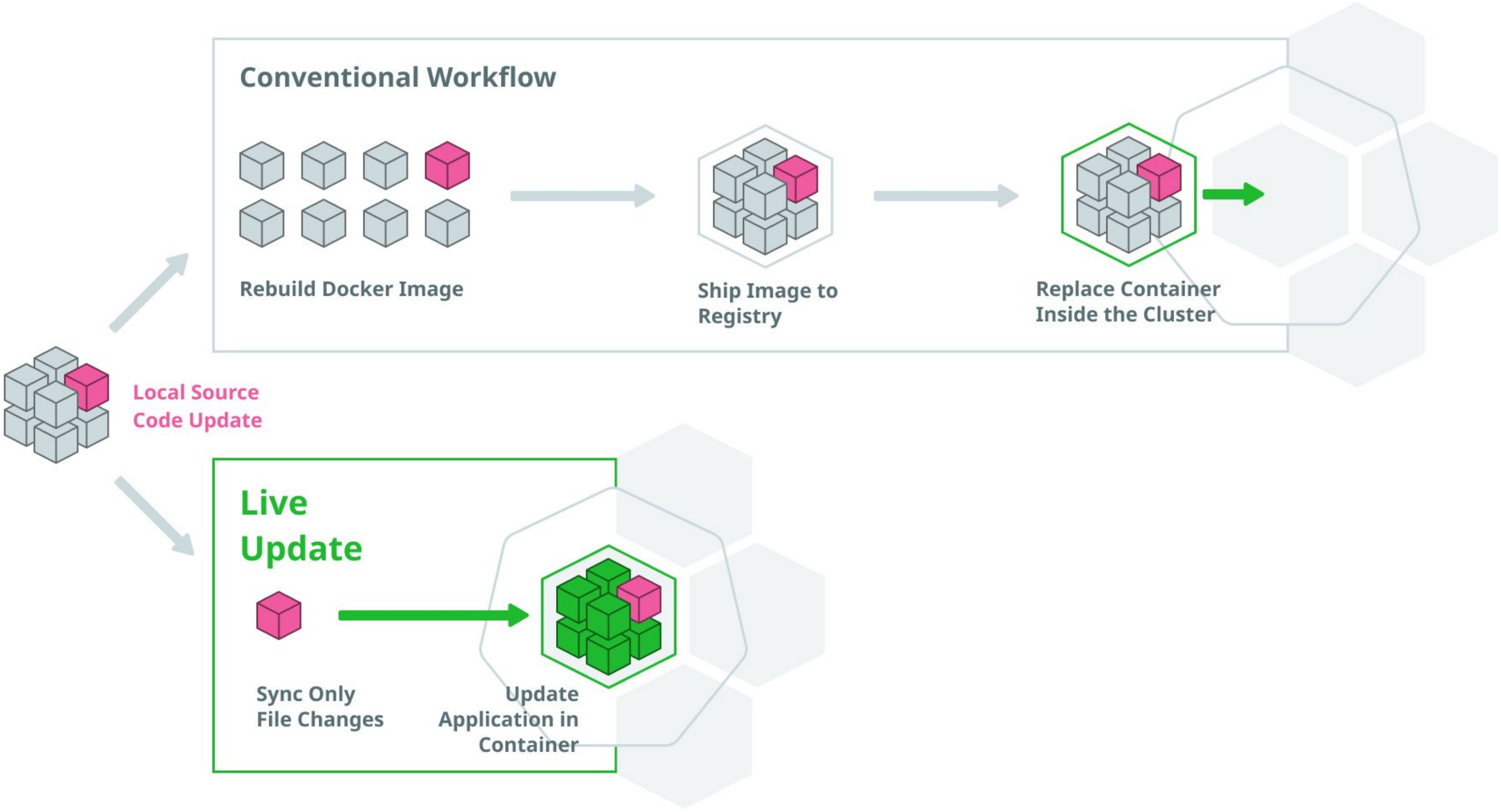
1,931 votes · Final results

3:38 PM · Nov 29, 2019 · Twitter Web App manual app update == 

Why?

Developer cognitive load:

- Developers like to **stay focused**
  - **# of operations** per code change?
  - **Time** from change to new process?
- Custom workflow automation
- Onboarding



Conventional Workflow

Rebuild Docker Image

Ship Image to
Registry

Replace Container
Inside the Cluster

Local Source
Code Update

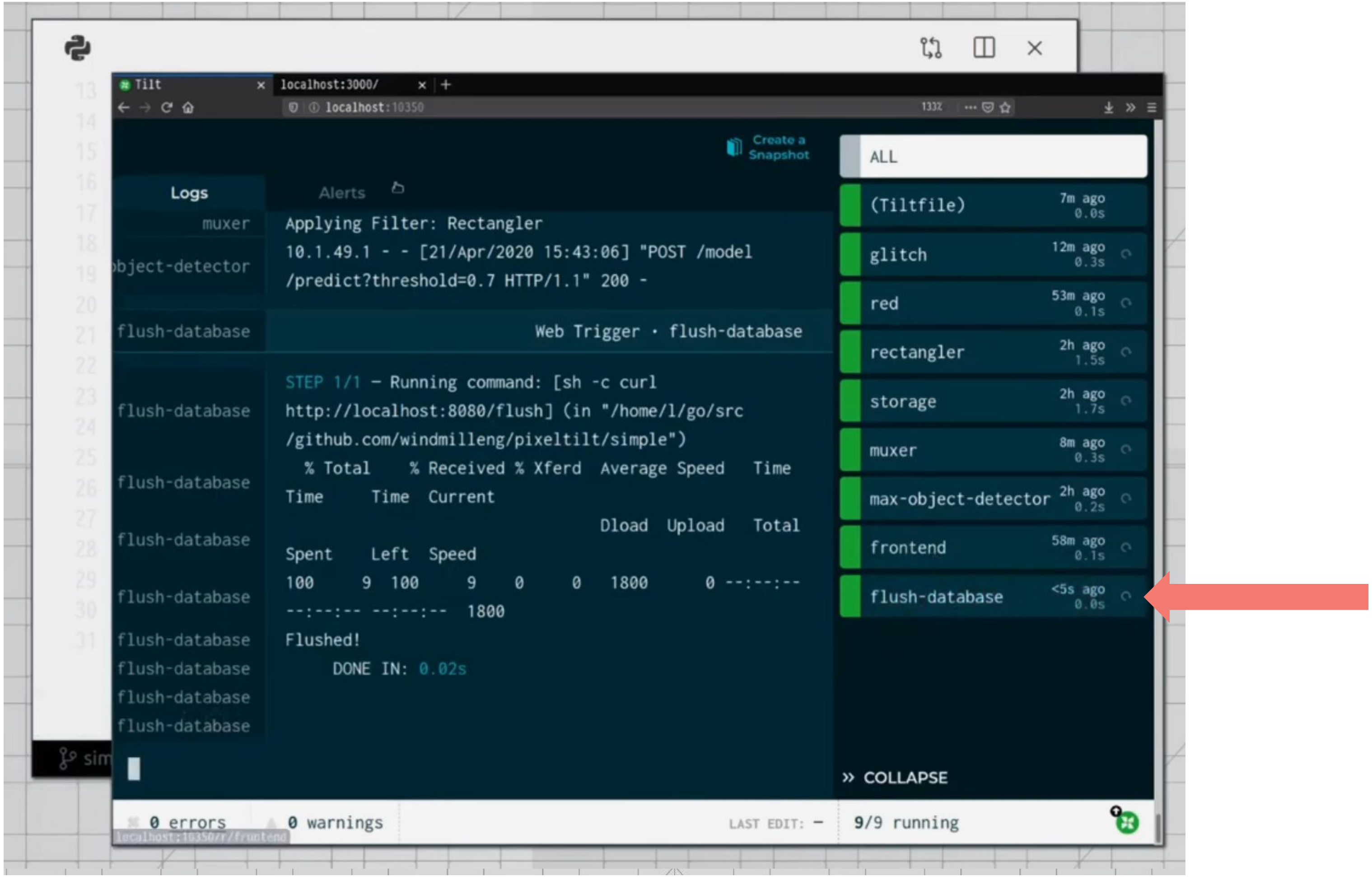
Live
Update

Sync Only
File Changes

Update
Application in
Container

Datadog

- Rolling out **Tilt**, currently at ~40%.
- **CI image** pulled locally
- **Build locally** inside CI image
- Tilt wraps Helm
- Easily discoverable **buttons in Tilt:**
  - Get dependencies
  - DB migrations



```
          muxer  Applying Filter: Rectangler
                 10.1.49.1 - - [21/Apr/2020 15:43:06] "POST /model
bject-detector   /predict?threshold=0.7 HTTP/1.1" 200 -

flush-database                      Web Trigger • flush-database

                 STEP 1/1 — Running command: [sh -c curl
flush-database   http://localhost:8080/flush] (in "/home/l/go/src
                 /github.com/windmilleng/pixeltilt/simple")
                   % Total    % Received % Xferd  Average Speed   Time
flush-database   Time     Time  Current
                                                  Dload  Upload   Total
flush-database   Spent    Left  Speed
                 100      9  100      9    0     0   1800      0 --:--:--
flush-database   --:--:-- --:--:--  1800
flush-database   Flushed!
flush-database        DONE IN: 0.02s
flush-database
flush-database
```

@ellenkorbes

unu

- unu inspired Tilt’s **extensions** feature!
- **Tilt + tons of automation**, such as:

unu

- unu inspired Tilt’s **extensions** feature!
- **Tilt + tons of automation**, such as:

- Internal Traefik proxy
- TLS management
- Vault integration
- Tracing support
- Sharded, replicated mongo cluster
- Prometheus alerts
- Live reload for Grafana dashboards (!)
- *Special thanks: David Rubin, who wrote the first third-party Tilt extension!*

Mindspace

Mindspace:

- **Tilt**, specifically for dev ↔ prod parity
- Had tons of Tilt **hacks that** eventually became **native features**

Cluster API

- User-specific **Tilt** settings on tilt-settings.json **overlays** on top of **defaults**
- Very complex Tilt **automation** e.g. cert. management functionality
- Used to build the Go binary in the container, with a full toolchain in the dev image, now **building binaries locally**

Takeaways

- Pattern:
  - Uniform **services** fit a common **structure**, and allow for recycling configs, live reload settings, etc
  - DevEx teams **automate everything** e.g. unu’s service discovery, Traefik, etc.

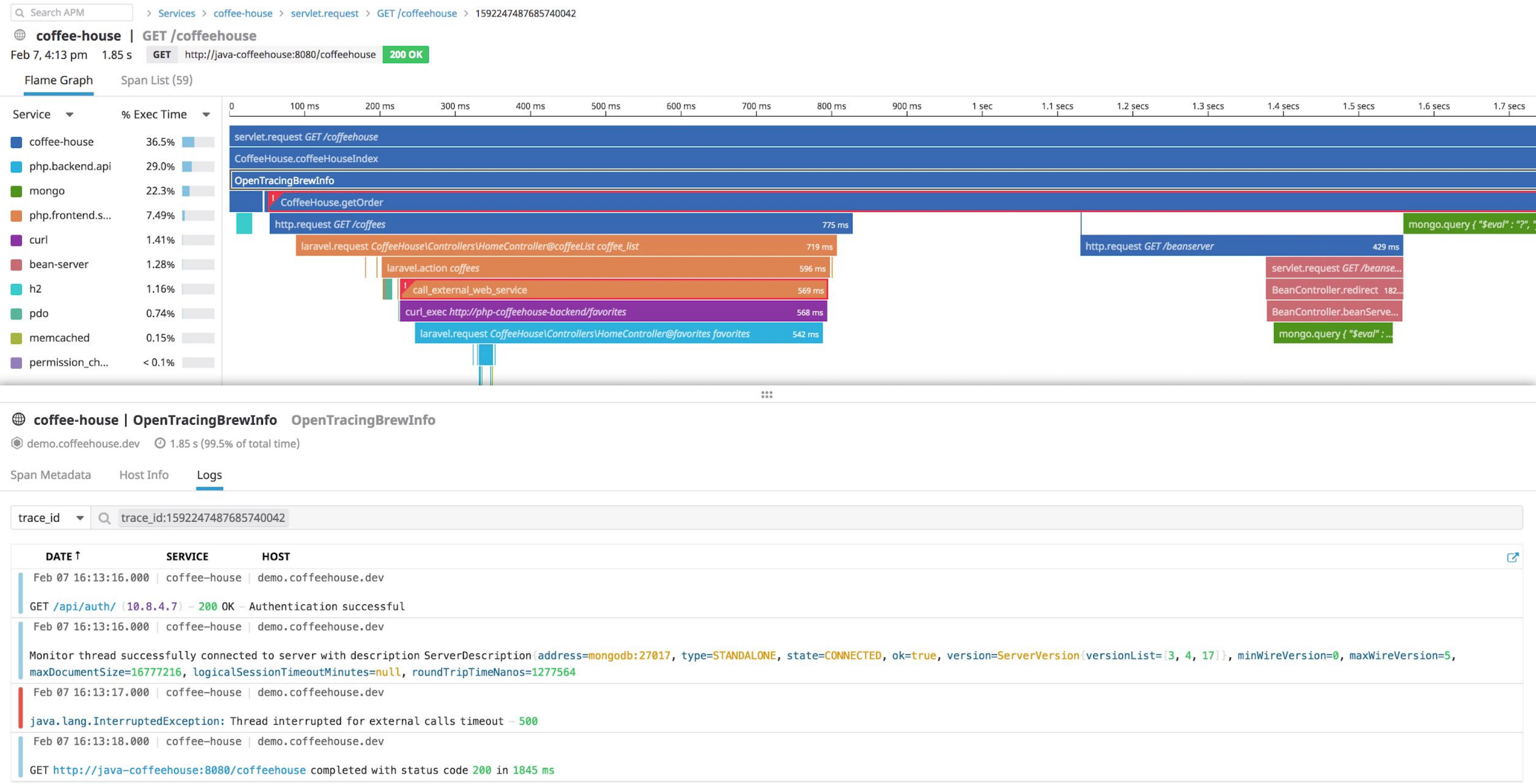
The Problem

Problem Solving

Single Player

Datadog & unu

- No debuggers. Output to **logs**, then use **tracing**.
- Datadog uses Datadog for metrics & traces
- **Auto-instrumentation** helps!
- unu uses Jaeger



Search APM > Services > coffee-house > servlet.request > GET /coffeehouse > 1592247487685740042

coffee-house | GET /coffeehouse

Feb 7, 4:13 pm 1.85 s GET http://java-coffeehouse:8080/coffeehouse 200 OK

Flame Graph Span List (59)

Service	% Exec Time
coffee-house	36.5%
php.backend.api	29.0%
mongo	22.3%
php.frontend.s...	7.49%
curl	1.41%
bean-server	1.28%
h2	1.16%
pdo	0.74%
memcached	0.15%
permission_ch...	< 0.1%

0
100 ms
200 ms
300 ms
400 ms
500 ms
600 ms
700 ms
800 ms
900 ms
1 sec
1.1 secs
1.2 secs
1.3 secs
1.4 secs
1.5 secs
1.6 secs
1.7 secs
servlet.request GET /coffeehouse
CoffeeHouse.coffeeHouseIndex
OpenTracingBrewInfo
CoffeeHouse.getOrder
http.request GET /coffees 775 ms
mongo.query { "$eval" : "?", "
laravel.request CoffeeHouse\Controllers\HomeController@coffeeList coffee_list 719 ms
http.request GET /beanserver 429 ms
laravel.action coffees 596 ms
servlet.request GET /beanse...
call_external_web_service 569 ms
BeanController.redirect 182...
curl_exec http://php-coffeehouse-backend/favorites 568 ms
BeanController.beanServe...
laravel.request CoffeeHouse\Controllers\HomeController@favorites favorites 542 ms
mongo.query { "$eval" : ...

coffee-house | OpenTracingBrewInfo OpenTracingBrewInfo

demo.coffeehouse.dev 1.85 s (99.5% of total time)

Span Metadata Host Info Logs

trace_id trace_id:1592247487685740042

```
DATE ↑                  SERVICE         HOST
Feb 07 16:13:16.000 | coffee-house | demo.coffeehouse.dev

GET /api/auth/ (10.8.4.7) - 200 OK - Authentication successful

Feb 07 16:13:16.000 | coffee-house | demo.coffeehouse.dev

Monitor thread successfully connected to server with description ServerDescription{address=mongodb:27017, type=STANDALONE, state=CONNECTED, ok=true, version=ServerVersion{versionList=[3, 4, 17]}, minWireVersion=0, maxWireVersion=5,
maxDocumentSize=16777216, logicalSessionTimeoutMinutes=null, roundTripTimeNanos=1277564

Feb 07 16:13:17.000 | coffee-house | demo.coffeehouse.dev

java.lang.InterruptedException: Thread interrupted for external calls timeout - 500

Feb 07 16:13:18.000 | coffee-house | demo.coffeehouse.dev

GET http://java-coffeehouse:8080/coffeehouse completed with status code 200 in 1845 ms
```

Mindspace & Cluster API

Mindspace:

- **Remote debugging**: IDE connects to node remote debugging; Tilt exposes the ports

Cluster API:

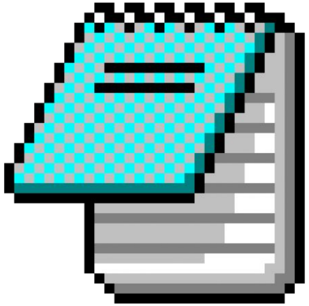
- Debugging: **Printlines**—no support for debuggers
- **Quick feedback loop** means this is fine

Takeaway

- Integration with **tracing** (Datadog & unu) and **debugging** tools (Mindspace) is still rare but growing!

Problem Solving

Multi Player

copy&paste logs == 

Datadog

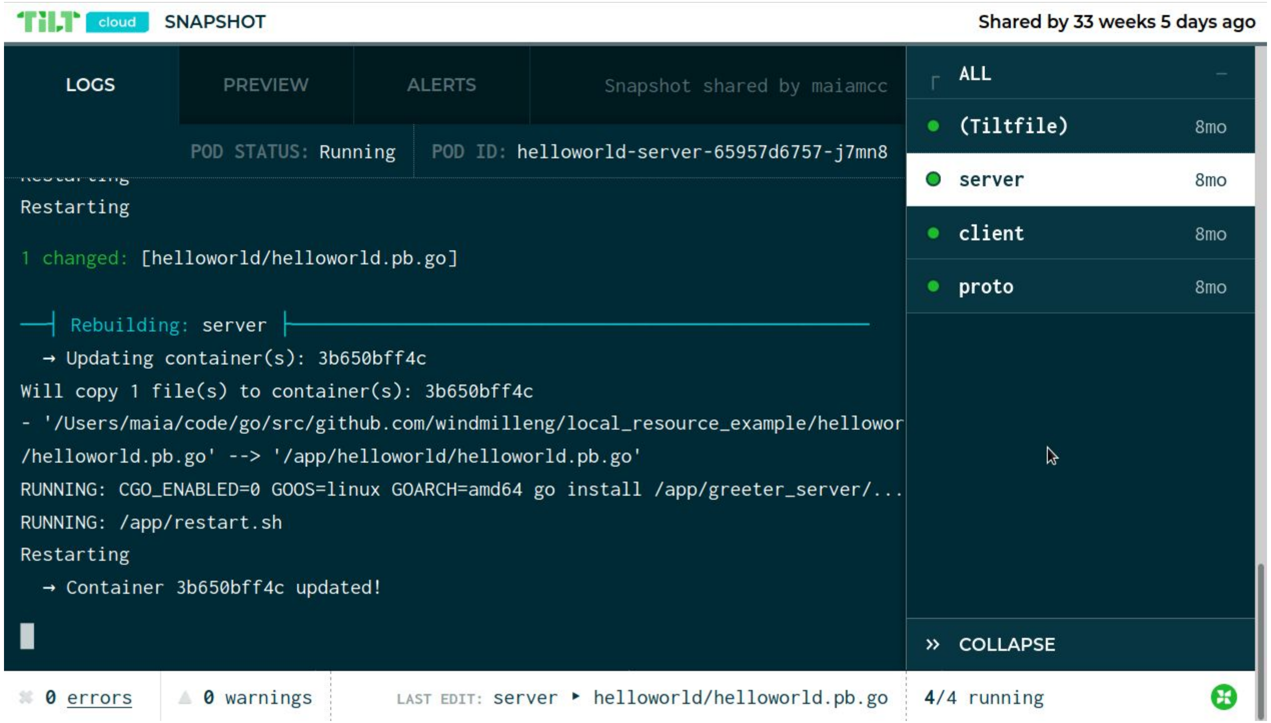
- Devs work in public, shareable namespaces
- Use wrapper tool to switch namespaces
- **Any dev can access another dev's namespace**

Mindspace

- No need for high-tech when it’s a small team sharing the same office!

Cluster API

- Snapshots!



Shared by 33 weeks 5 days ago

Snapshot shared by maiamcc

POD STATUS: Running   POD ID: helloworld-server-65957d6757-j7mn8

```
Restarting
Restarting

1 changed: [helloworld/helloworld.pb.go]

──┤ Rebuilding: server ├──────────────────────────────────
  → Updating container(s): 3b650bff4c
Will copy 1 file(s) to container(s): 3b650bff4c
- '/Users/maia/code/go/src/github.com/windmilleng/local_resource_example/hellowor
/helloworld.pb.go' --> '/app/helloworld/helloworld.pb.go'
RUNNING: CGO_ENABLED=0 GOOS=linux GOARCH=amd64 go install /app/greeter_server/...
RUNNING: /app/restart.sh
Restarting
  → Container 3b650bff4c updated!
```

0 errors   0 warnings   LAST EDIT: server ▸ helloworld/helloworld.pb.go   4/4 running

Takeaways

- **Big** companies: You have a **namespace** and your colleague logs into it
- **Middle** ground: **Snapshots**
- **Small** companies: analog solutions

The Future

Custom Workflows

Custom Workflows

- We’ve learned there’s no ideal workflow

Custom Workflows

- We’ve learned there’s no ideal workflow
- Everyone needs their own setup

Custom Workflows

- We’ve learned there’s no ideal workflow
- Everyone needs their own setup
- Kubernetes won because it’s flexible

Custom Workflows

- We’ve learned there’s no ideal workflow
- Everyone needs their own setup
- Kubernetes won because it’s flexible
- Helm/Kustomize integration

Custom Workflows

- We've learned there's no ideal workflow
- Everyone needs their own setup
- Kubernetes won because it's flexible
- Helm/Kustomize integration
- Datadog: Buttons! Bazel!

Custom Workflows

- We’ve learned there’s no ideal workflow
- Everyone needs their own setup
- Kubernetes won because it’s flexible
- Helm/Kustomize integration
- Datadog: Buttons! Bazel!
- ClusterAPI: Cert. manager! User overrides!

Custom Workflows

- We've learned there's no ideal workflow
- Everyone needs their own setup
- Kubernetes won because it's flexible

- Helm/Kustomize integration
- Datadog: Buttons! Bazel!
- ClusterAPI: Cert. manager! User overrides!
- Unu: Sharded Mongo cluster!

Custom Workflows

- We've learned there's no ideal workflow
- Everyone needs their own setup
- Kubernetes won because it's flexible

- Helm/Kustomize integration
- Datadog: Buttons! Bazel!
- ClusterAPI: Cert. manager! User overrides!
- Unu: Sharded Mongo cluster!
- ...and dozens of user-contributed Tilt extensions.

Custom Workflows

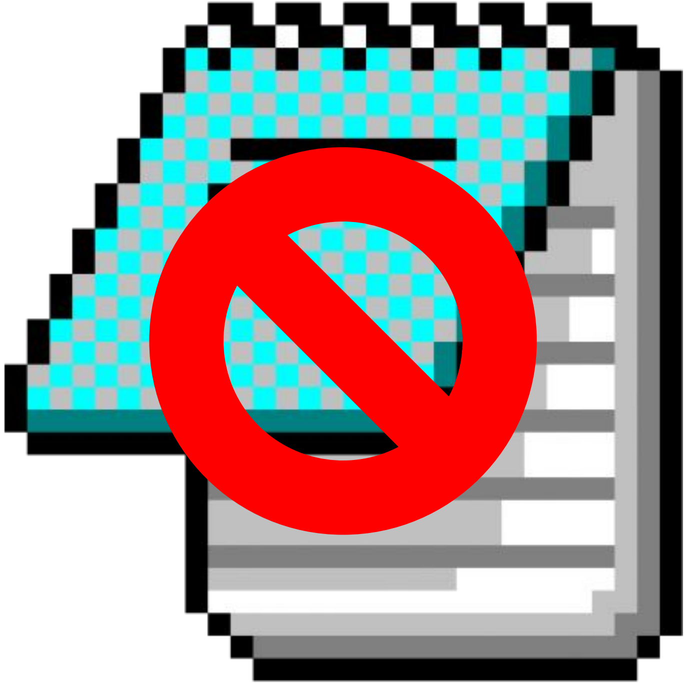
- N tools means N^2 combinations

Custom Workflows

- N tools means N^2 combinations
- It’s a lot to maintain

Custom Workflows

- N tools means N^2 combinations
- It’s a lot to maintain
- **We need to do for Kubernetes what the IDE did in the late 80s.**



Successful Kubernetes Development Workflows

Thank You!

Ellen Körbes

Head of Product

- l@tilt.dev
- @ellenkorbes
- they/she
- #tilt@slack.k8s.io

Featured:

Datadog	datadoghq.com
unu	unumotors.com
Mindspace	mindspace.net
Cluster API	cluster-api.sigs.k8s.io