
KubeCon
CloudNativeCon
Europe 2023
TiKV

1M Lines of YAML: Wrangling Kubernetes Configuration for Hundreds of Teams

Katrina Verey, Shopify

About me

- Senior Staff Developer, Production Engineering @ Shopify
- Kubernetes end user and enthusiast since 2016
- Upstream lead since 2021:
  - SIG CLI Tech Lead & Chair
  - Kustomize subproject owner
  - KRM Functions subproject owner

About this talk

✓ Managing Kubernetes YAML at scale

1500+ app configuration sets ~> 1M+ lines of YAML

Far more than that in our 350+ production clusters

✓ Platform Engineering principles

✗ Specific config management tools

✓ Systems those tools are part of

YAML

The Problem:
~~YAML~~ Intent Management

Declarative APIs 💖

"Kubernetes is not just API-driven, but is ***API-centric***."

- Brian Grant, "The Kubernetes Resource Model (KRM)", 2018 (https://git.k8s.io/design-proposals-archive/architecture/resource-management.md)

Declarative APIs 💖

"In Kubernetes, **declarative abstractions are primary**"

- Brian Grant, "The Kubernetes Resource Model (KRM)", 2018 (https://git.k8s.io/design-proposals-archive/architecture/resource-management.md)

"A simple foundation for a complex space,
Desired state empowers with effortless grace."

- ChatGPT

Starting a platform

App init

Stack

- (•) Rails (default)
- ( ) Go

Select the deployments that are needed for your application.
Each selection will be included in a pull request on your service's repo.
You'll have to review this pull request to complete your runtime setup.
For more information visit the docs.

- [ ] kube-shell
- [ ] elasticsearch
- [ ] web
- [ ] db-migrate
- [ ] asset-uploading
- [ ] jobs
- [ ] redis
- [ ] cloudsql
- [ ] memcached

Per-commit variance

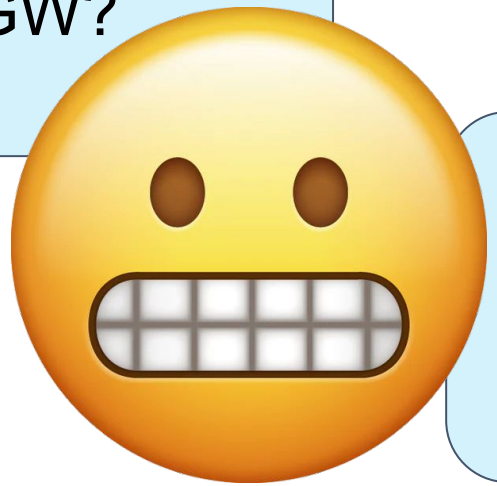

```yaml
spec:
  containers:
  - name: web
    image: "registry.shopify.io/path/to/shopify-app:2f2a598a302060945bda4ffc70b23d6677597936"
    env:
    - name: REVISION
      value: "2f2a598a302060945bda4ffc70b23d6677597936"
```

Solution: ERB


```ruby
def apply_all_templates
  found = 0
  Dir.foreach(@template_path) do |file|
    file_path = "#{@template_path}/#{file}"
    if File.extname(file) == '.yml'
      found += 1
      run_command('kubectl', 'apply', '-f', file_path, "--namespace=#{@namespace}")
    elsif File.extname(file) == '.erb'
      found += 1
      render_and_apply_template(file_path)
    end
  end
  raise FatalDeploymentError, "No templates found in #{@template_path}" if found.zero?
end
```

ERB: A big mistake

Hmm, a Turing-complete language in config files! WCGW?



It's SO convenient for everyone! I'm sure it'll be fine.

My very smart boss

2016 Me

Why it was bad

- As predicted, people did all sorts of crazy things with it
  - Pseudo-abstractions? 👌
  - Complex math? 👌
  - HTTP calls!?!? 👌

Why it was bad

- We couldn't parse it!
  - String scanning as a maintenance strategy 👎👎👎
  - Delayed updates & unintended drift

- Missing interface
  - No separation of concerns
  - Customizations only identifiable via git
  - High learning curve for devs

Deploy-time rendering: a worse mistake!

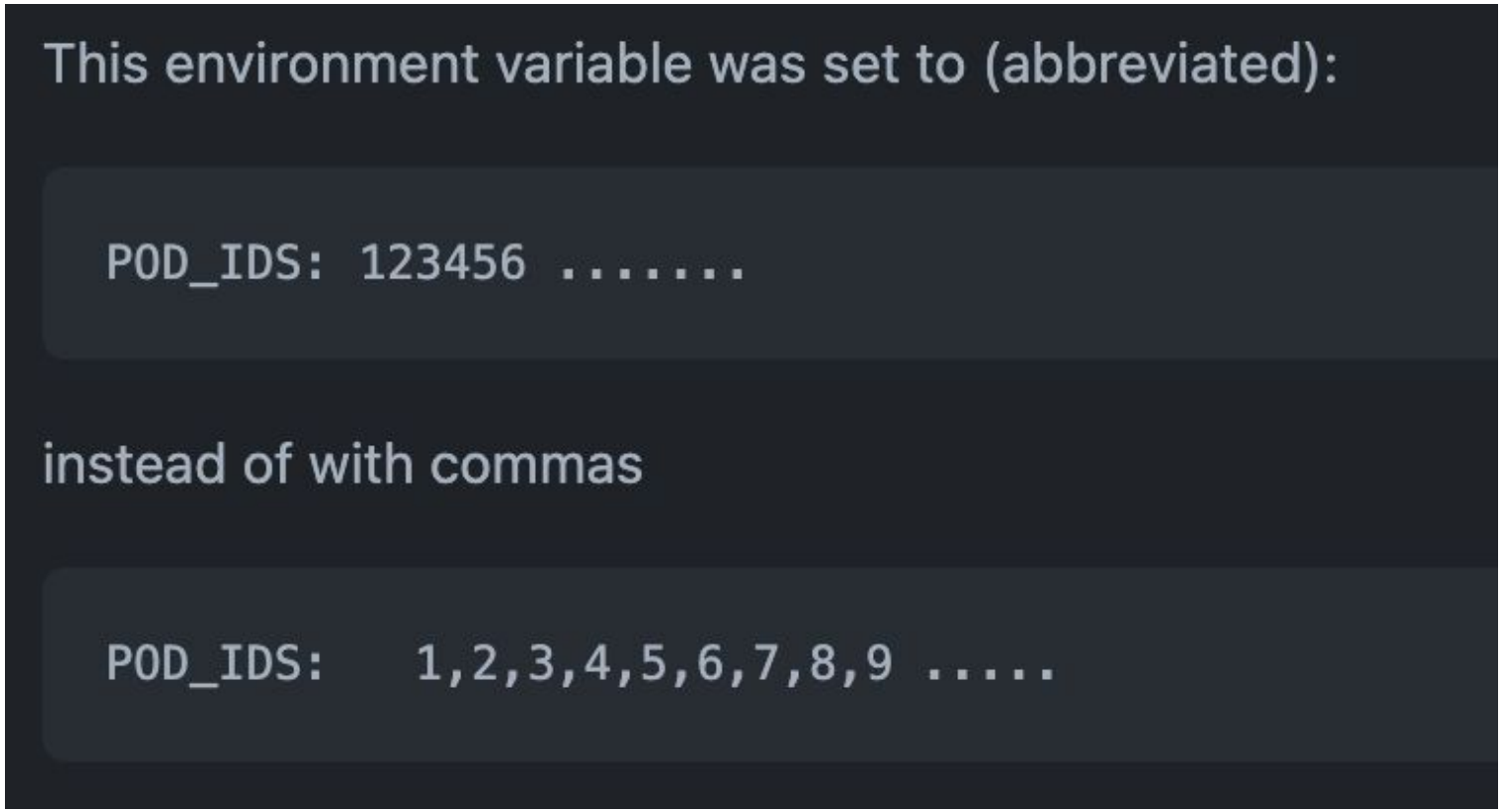
KubeCon CloudNativeCon Europe 2023

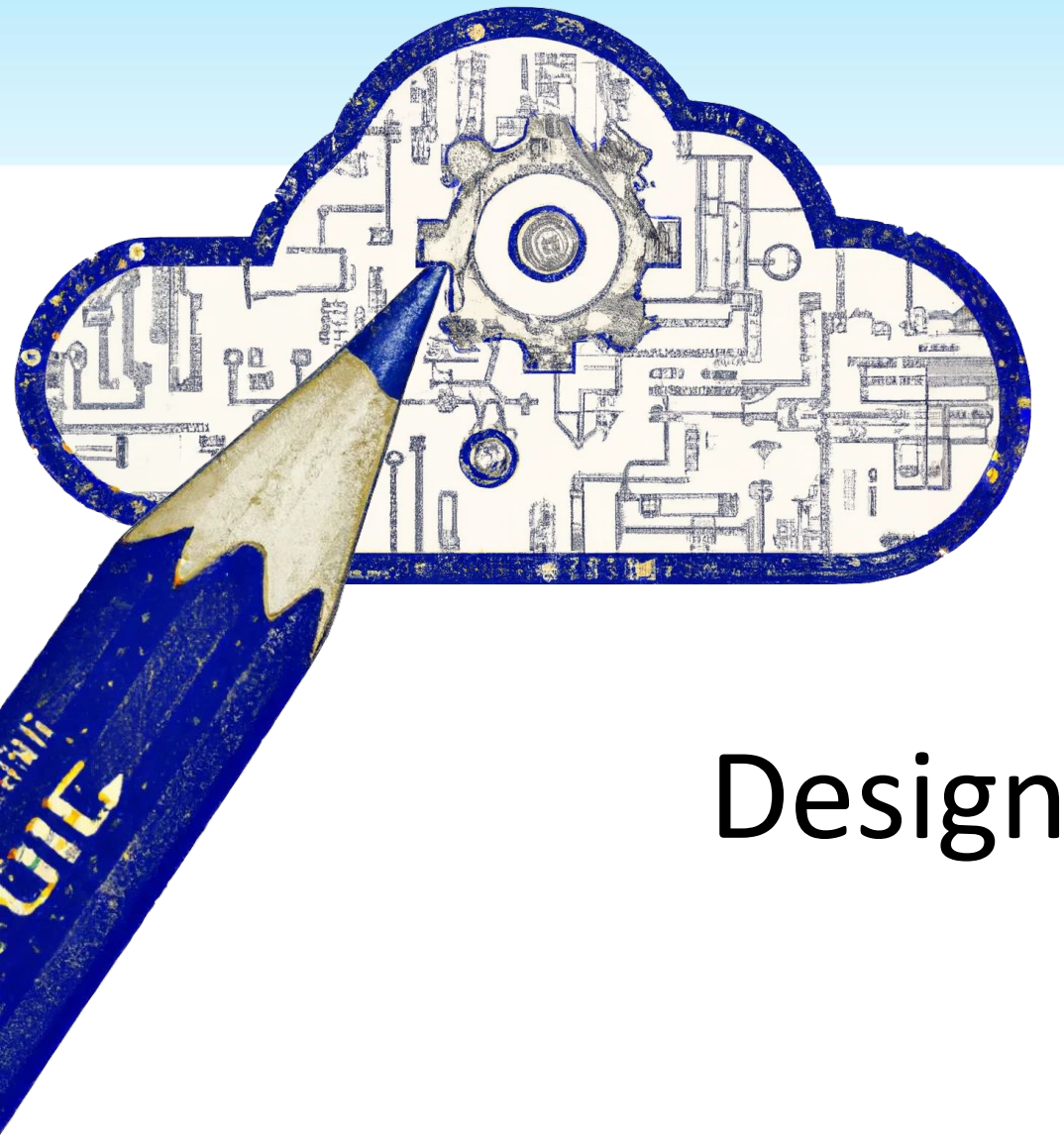
Unintended changes can be hard to catch

```
      1  + <% %w(web web-perf).each do |name| %>
1     2    apiVersion: v1
2     3    kind: Service
3     4    metadata:
4          -   name: web
      5  +   name: <%= name %>
5     6    annotations:
           @@ -231,6 +232,7 @@ spec:
231   232              - name: star-shopifyio
232   233                secret:
233   234                  secretName: star-shopifyio
      235  + <% end %>
234   236    ---
```

Deploy-time rendering: a worse mistake!

Rollbacks were reliant on rendering





Designing a solution

Desired properties

1. **Safety:**
    a. Deploys AND rollbacks must be **repeatable**.
    b. ALL changes must be **easily reviewable**

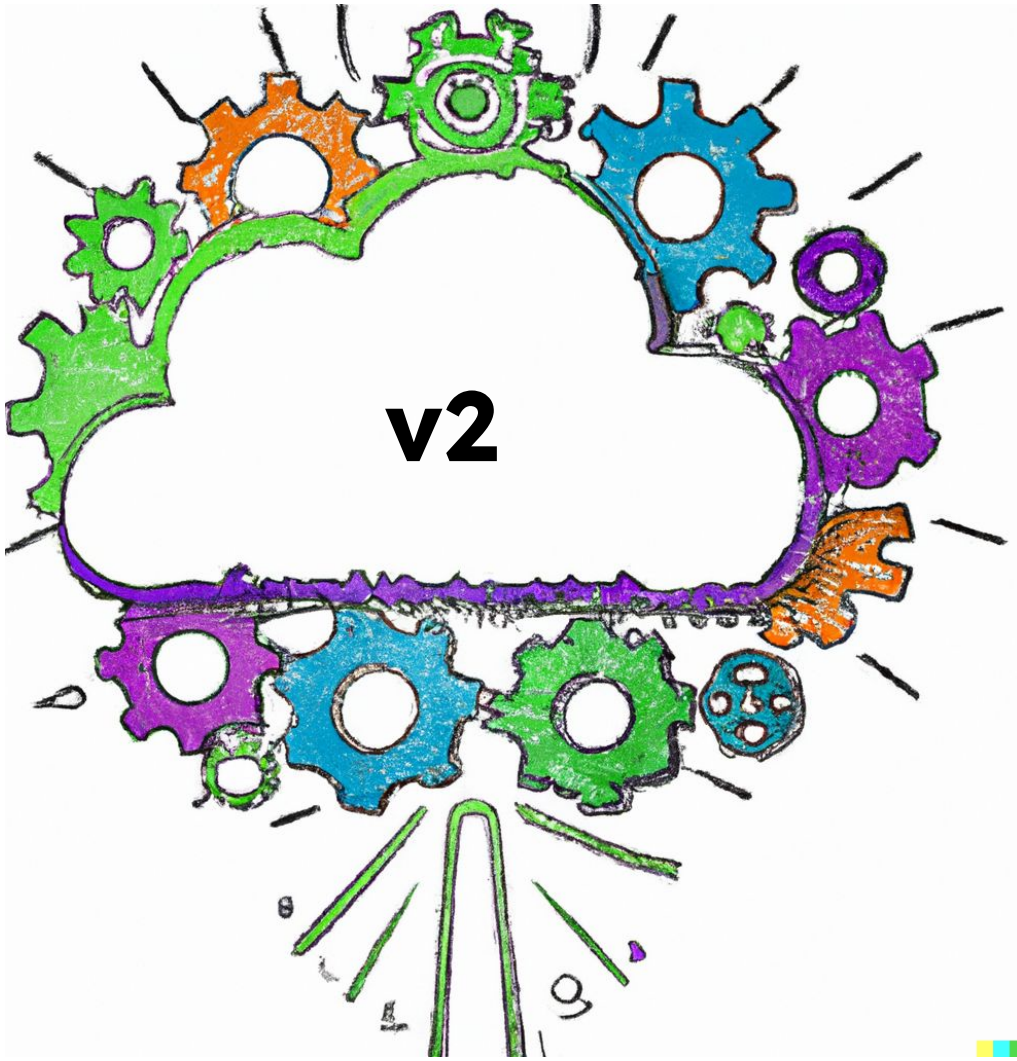
2. Automation-readiness
3. Universality

Desired properties

1. Safety

2. **Automation-readiness:**
   a. Possible to **programmatically** make arbitrary updates **reliably**
   b. Possible to evolve practices through **versioning**

3. Universality



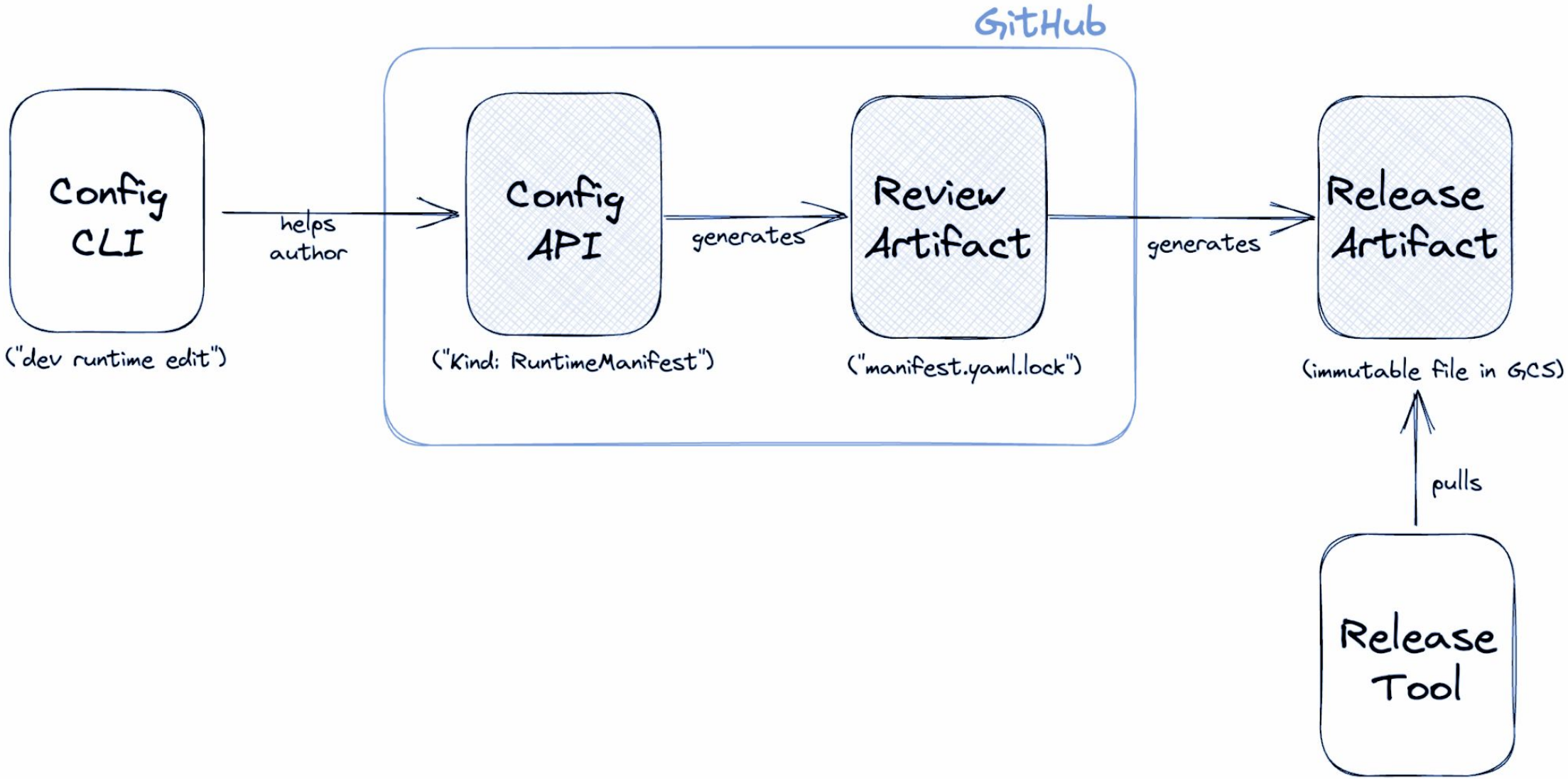
Desired properties

1. Safety
2. Automation-readiness

3. **Universality:**
    a. Simple to start with
    b. Simple to make common modifications
    c. Possible to make arbitrary customizations as needed

What did we build?

Solution overview



Config CLI


```
runtime add
Usage: runtime add [<runtime(s)>] [<component>]

  Add a component to one or more runtimes.
  Multiple runtimes can be manually set using a comma,separated,list for the runtime arg
```

```
runtime edit
Usage: runtime edit [<runtime>] [<component>]

  Change the configuration of a component for one or more runtimes
```

```
runtime upgrade
Usage: runtime upgrade [<runtime>] [<component>] [--status] [--latest] [--deprecated-versions] [--removed-versions]

  Upgrade a component for one or more runtimes to the latest version (or specified version).
  `all` can be used as an argument for <runtime> or <component> to specify all resource of that type.
```

Config API: `manifest.yaml`

KubeCon CloudNativeCon Europe 2023

```yaml
kind: RuntimeManifest
apiVersion: platform.shopify.io/v1alpha1
metadata:
  name: production-unrestricted
runtimeInfo:
  appName: shopify-app
  env: production
  appImage: registry.shopify.io/path/to/shopify-app
  domains:
  - shopify-app.shopify.io
components:
- name: web
```

Config API: manifest.yaml


```yaml
components:
- name: web
  basePath: web/v32
  resources:
  - base: ingress.yaml
  - base: certificate.yaml
    name: shopify-app-shopify-io
  - base: service.yaml
    patches:
    - github:Shopify/base_config_repo/patches/legacy/nginx/service.yaml
- name: job-worker
  resources:
  - base: jobs-deployment.yaml
    name: jobs
    patches:
    - repo_local/path/to/custom-labels.yaml
    - repo_local/path/to/rollout-strategy.yaml
```

Review artifact: `manifest.yaml.lock`


```
# THIS FILE IS AUTO-GENERATED. DO NOT MODIFY IT MANUALLY.
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jobs
  labels:
    name: jobs
    env: production
```

Review artifact: manifest.yaml.lock


```
97  -   basePath: web/v14
```

```
98  +   basePath: web/v22
```

```
847 -   activeDeadlineSeconds: 900
848     automountServiceAccountToken: false
849     containers:
850     - name: command-runner
        @@ -857,17 +857,30 @@ spec:
857         - configMapRef:
858             name: application-state
859         env:
```

```
847 +   activeDeadlineSeconds: 300
848     automountServiceAccountToken: false
849     containers:
850     - name: command-runner
857         - configMapRef:
858             name: application-state
859         env:
860 +     - name: STATSD_USE_NEW_CLIENT
861 +       value: "1"
862 +     - name: KUBE_POD_NAME
863 +       valueFrom:
864 +         fieldRef:
865 +           fieldPath: metadata.name
```

Release artifact


```
# THIS FILE IS AUTO-GENERATED. DO NOT MODIFY IT MANUALLY.
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jobs
  labels:
    name: jobs
    env: production
```

Release artifact

Config interface

```
5  runtimeInfo:
6    appName: shopify-app
7    env: production
8    appImage: registry.shopify.io/path/to/shopify-app
```

Review artifact

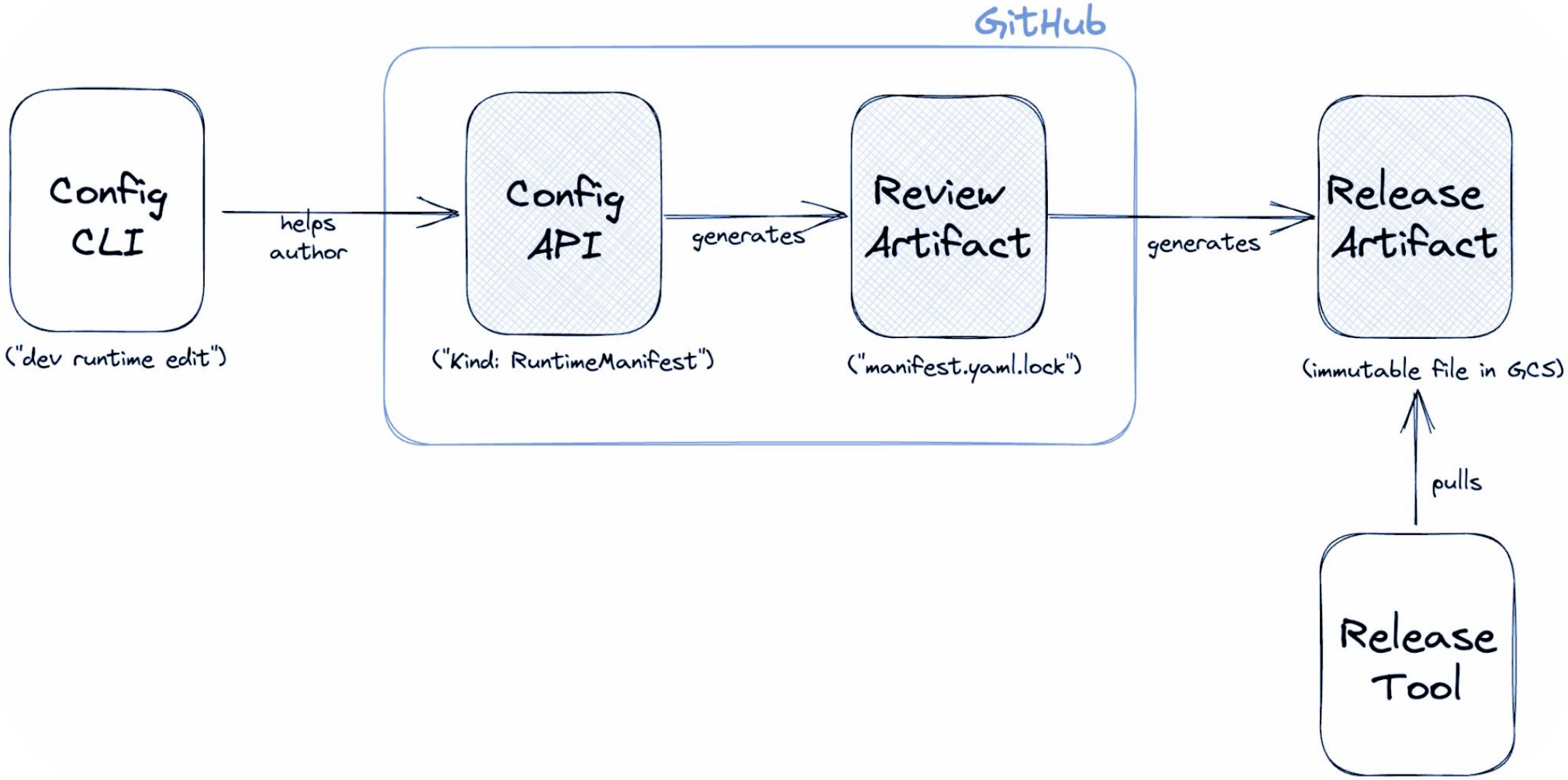
```
metadata:
  annotations:
    platform.shopify.io/app-containers: jobs
```

Release artifact

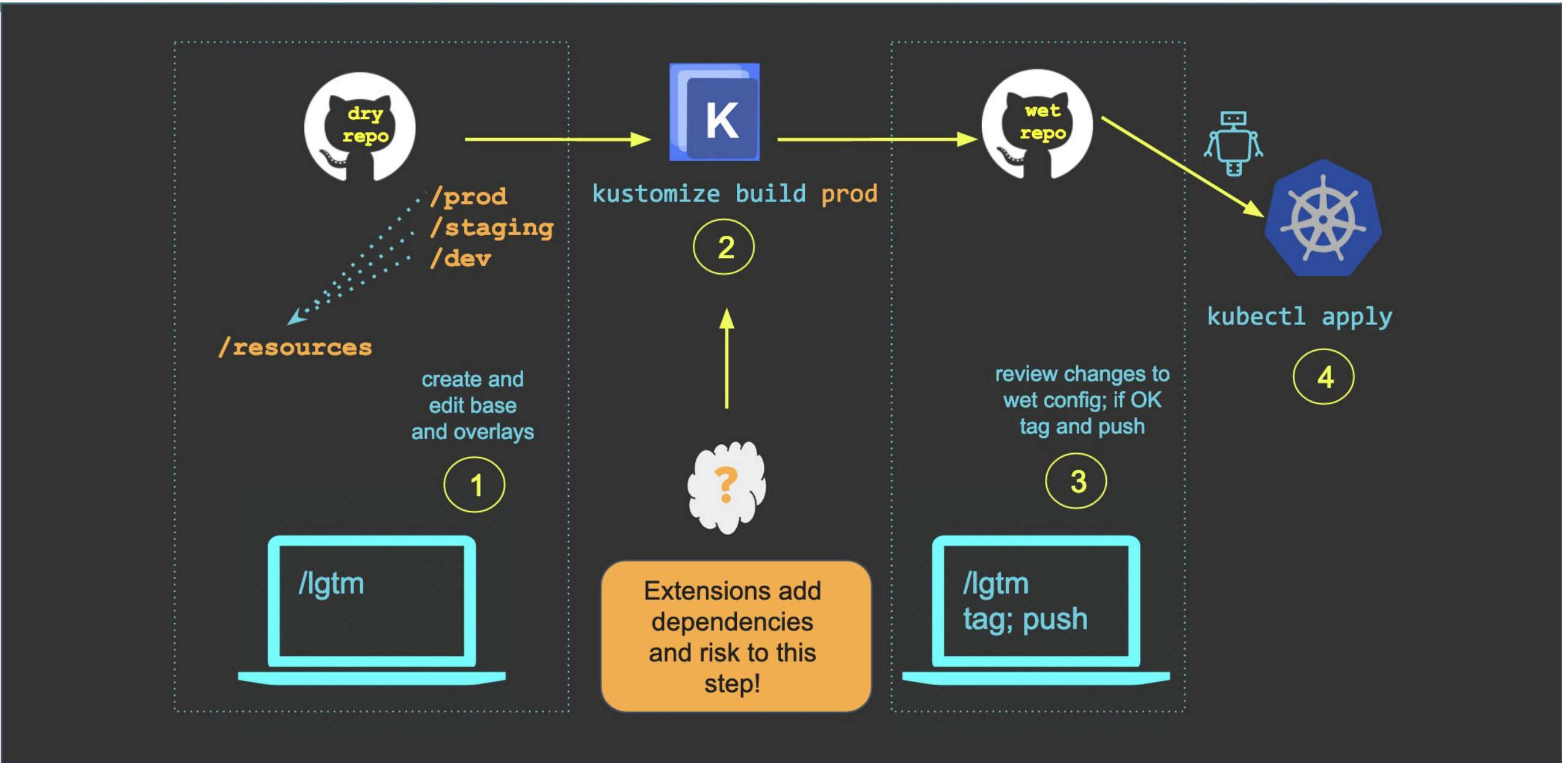
```
spec:
  containers:
  - name: jobs
    image: registry.shopify.io/path/to/shopify-app:2f2a598a302060945bda4ffc70b23d6677597936
```

What else could this look like?

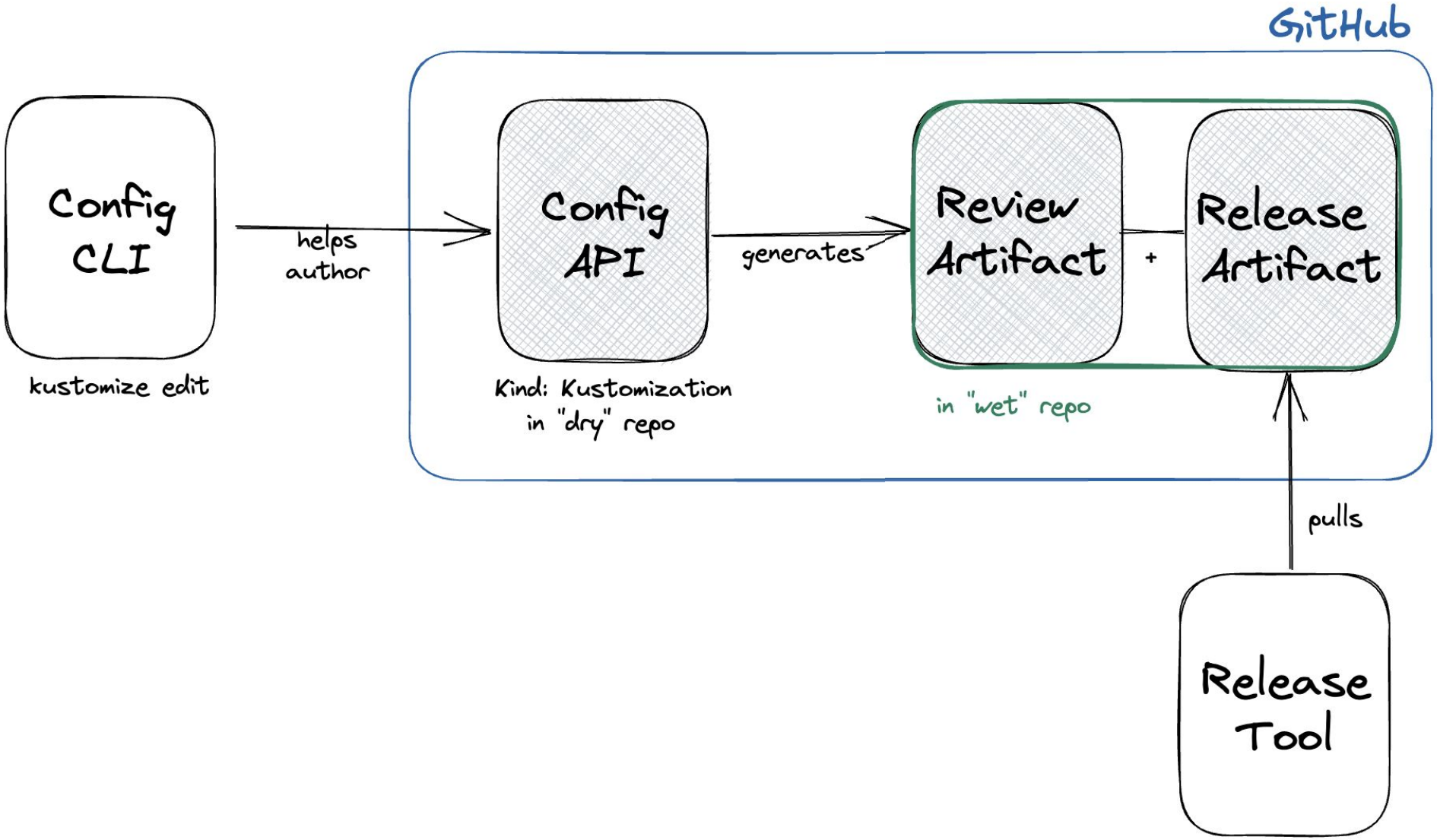
Alternatives



Example: Kustomize w/GitOps



Example: Kustomize w/GitOps



Why does this work well?

Principle 1:

Give your users a way to express *their* intent

Principle 2:

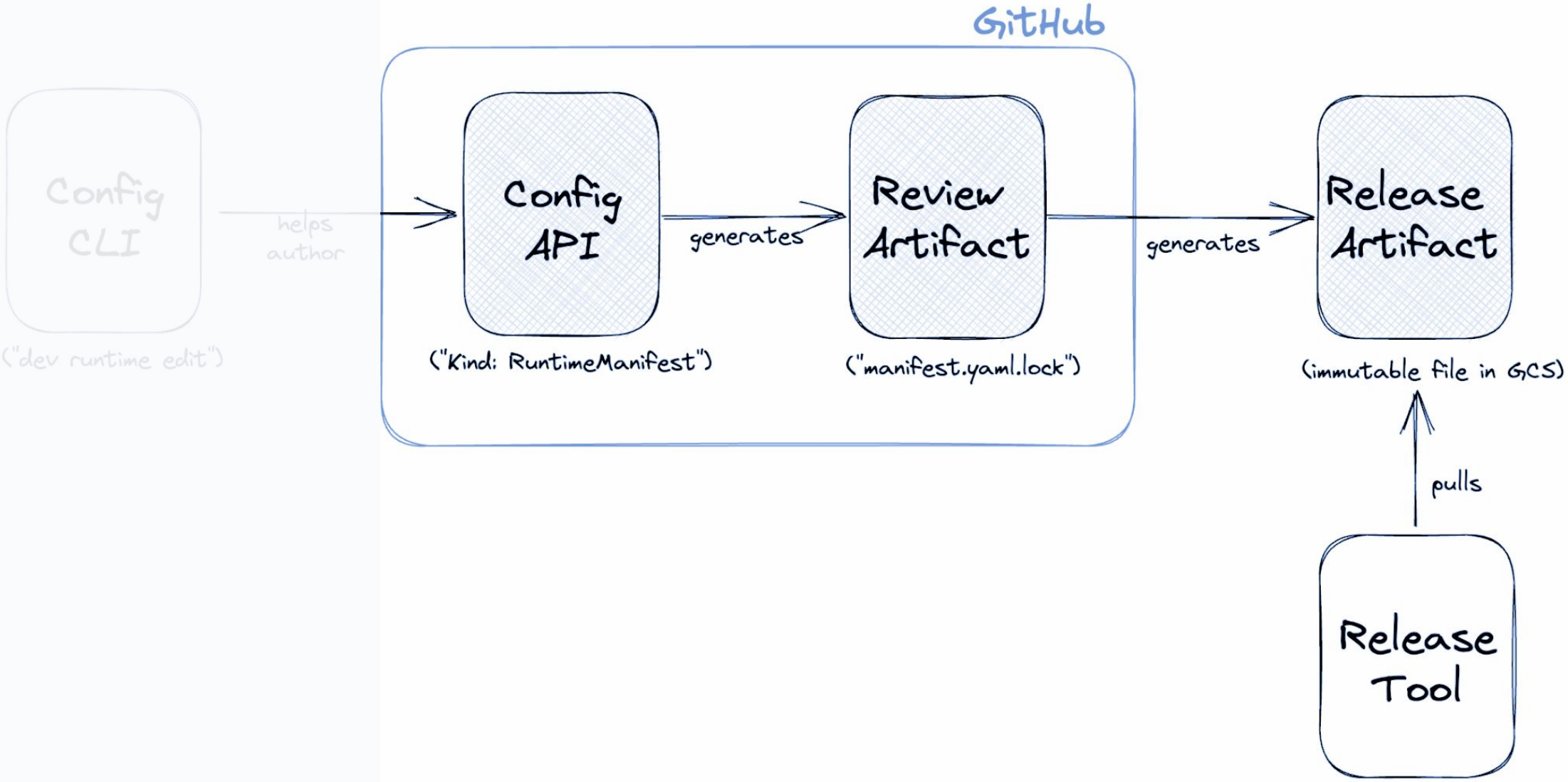
If you can't round-trip it,
don't distribute it

Principle 3:
Make consequential changes visible

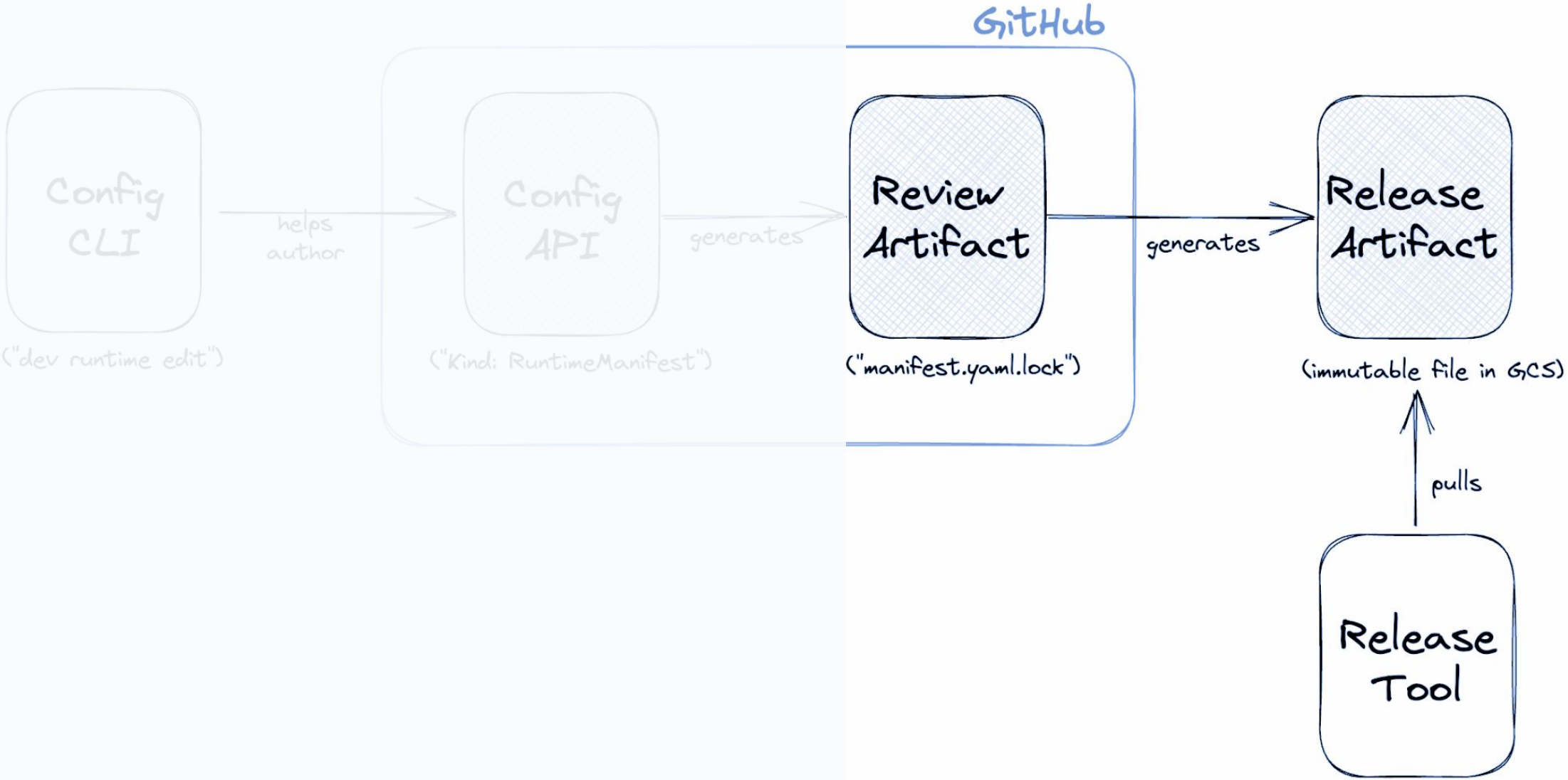
Principle 4:
Create release artifacts as early as possible

Principle 5:
Make it modular

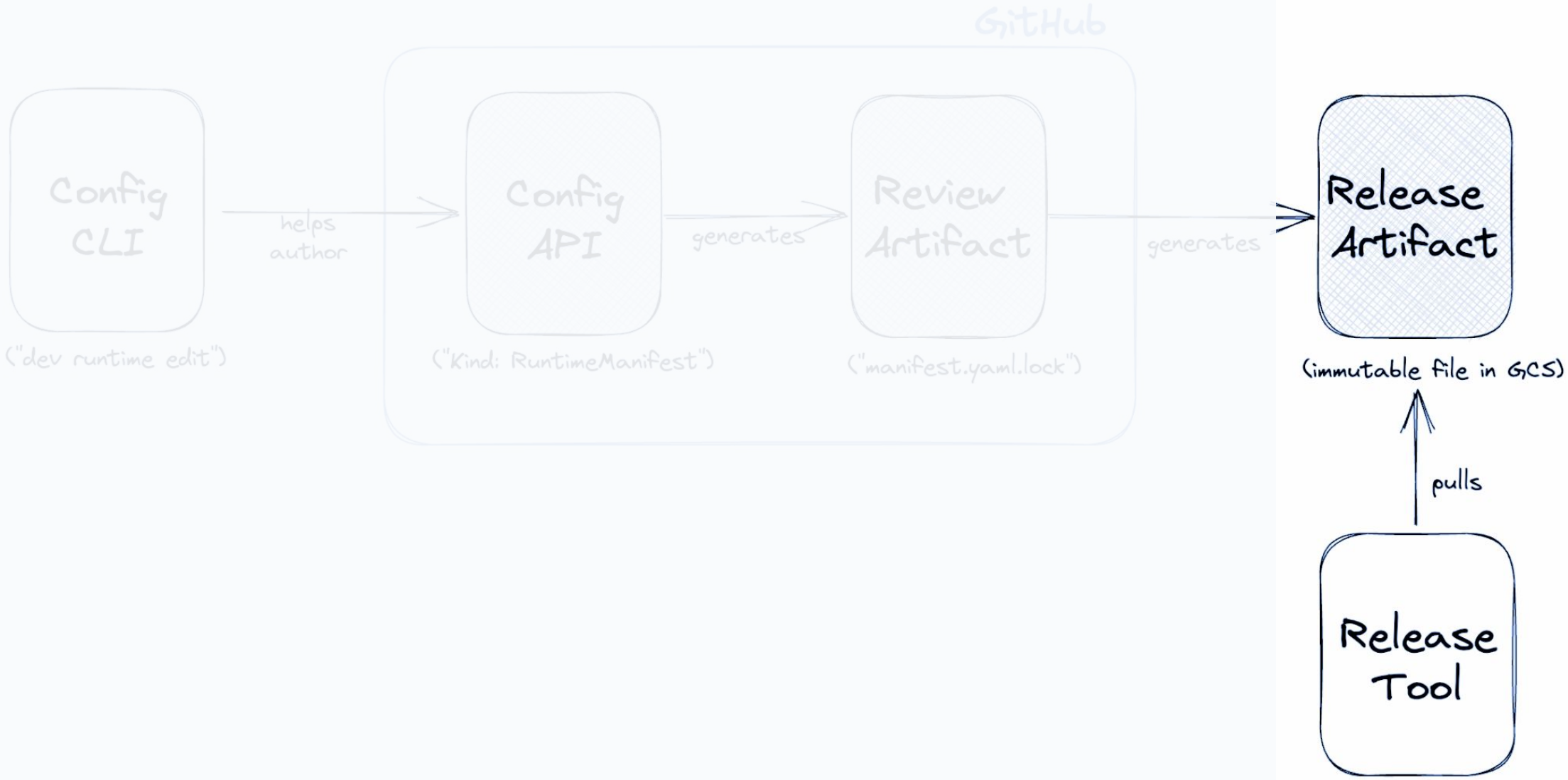
Modularity


GitHub
Config CLI
("dev runtime edit")
helps author
Config API
("Kind: RuntimeManifest")
generates
Review Artifact
("manifest.yaml.lock")
generates
Release Artifact
(immutable file in GCS)
pulls
Release Tool

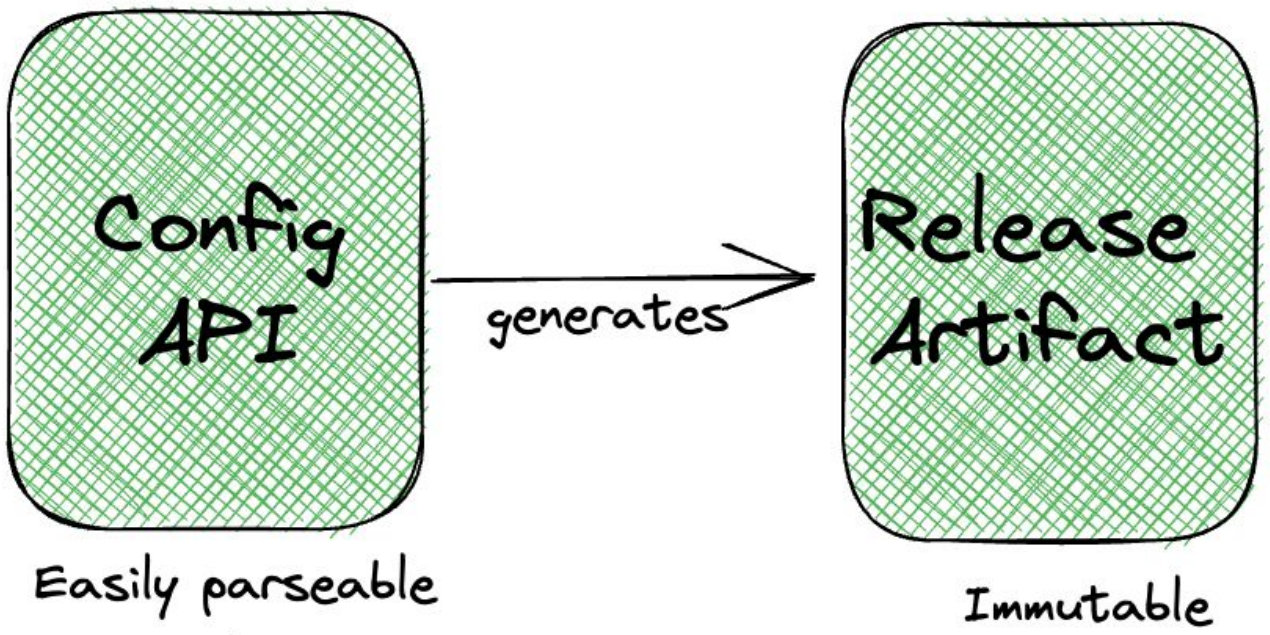
Modularity


Config CLI
("dev runtime edit")
helps author
Config API
("Kind: RuntimeManifest")
generates
GitHub
Review Artifact
("manifest.yaml.lock")
generates
Release Artifact
(immutable file in GCS)
pulls
Release Tool

Modularity


GitHub
Config CLI
helps author
("dev runtime edit")
Config API
("Kind: RuntimeManifest")
generates
Review Artifact
("manifest.yaml.lock")
generates
Release Artifact
(immutable file in GCS)
pulls
Release Tool

MVP


Config API
generates
Release Artifact
Easily parseable
Immutable

KRM Functions Framework: A Config API Toolkit

KRM what?

- **KRM**: Kubernetes Resource Model

```yaml
apiVersion: example.com/v1
kind: ExampleApp
metadata:
  name: my-app
```

- **KRM Function**: a piece of code that performs CRUD operations on *local* Kubernetes-style resources based on the desired state declared in a KRM resource.

KRM what?

- **KRM**: Kubernetes Resource Model
- **KRM Function**: a piece of code that performs CRUD operations on *local* Kubernetes-style resources based on the desired state declared in a local Kubernetes-style config API.
- **KRM Functions Specification**: a standard for the design of KRM functions that are small, interoperable, language-independent and chainable *as part of a pipeline that generates fully rendered configurations* that can be applied to a control plane.
- **KRM Functions Framework**: a toolkit for building specification-compliant KRM functions in Go

KRM what?

KRM Functions Framework: a toolkit for writing Go code that generates and/or modifies local Kubernetes configuration based on the desired state declared in a local Kubernetes-style API

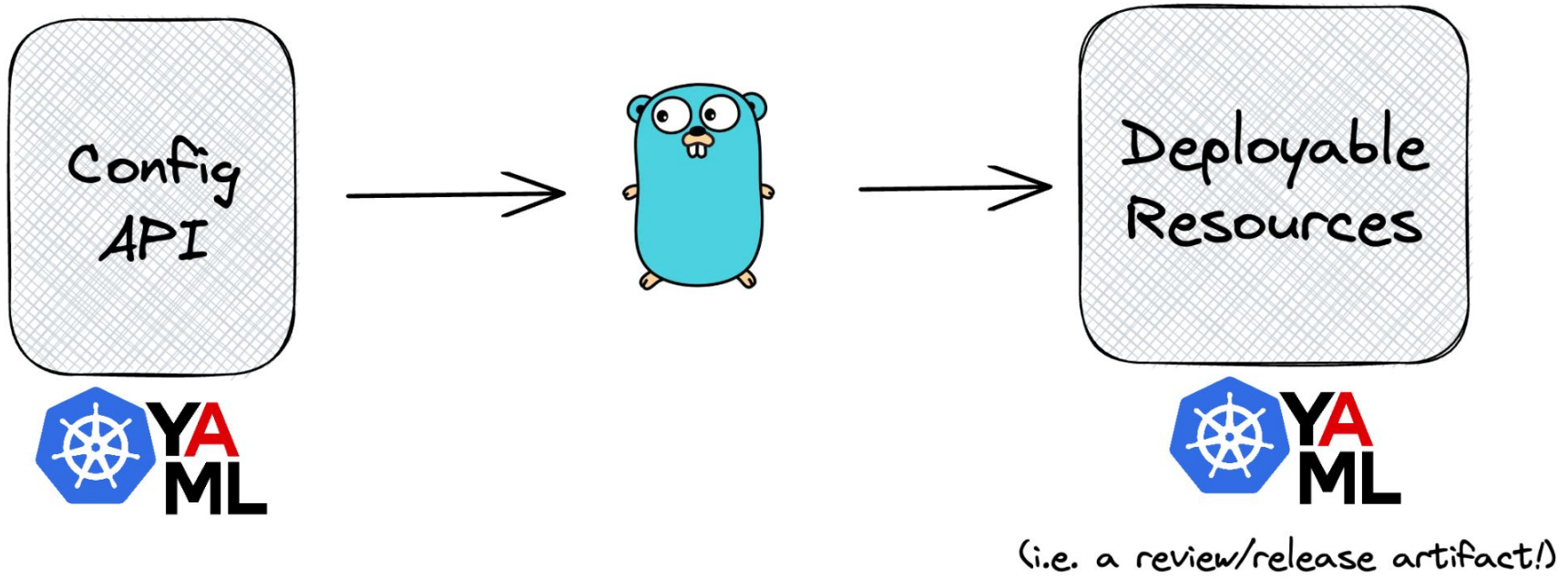


Image credit for the Go Gopher belongs to Renee French

KRM Functions Framework


```yaml
apiVersion: platform.example.com/v1alpha1
kind: ExampleApp
metadata:
  name: simple-app-sample
env: production
workloads:
  webWorkers:
    - name: web-worker
      domains:
      - example.com
```

KRM Functions Framework


```
26  type ExampleApp struct {
27          // Embedding these structs is required to use controller-gen to produce the CRD
28          metav1.TypeMeta   `json:",inline"`
29          metav1.ObjectMeta `json:"metadata"`
30
31          // +kubebuilder:validation:Enum=production;staging;development
32          Env string `json:"env" yaml:"env"`
33
34          // +optional
35          AppImage string `json:"appImage" yaml:"appImage"`
36
37          Workloads Workloads `json:"workloads" yaml:"workloads"`
38
39          // +optional
40          Datastores Datastores `json:"datastores,omitempty" yaml:"datastores,omitempty"`
41
42          // +optional
43          Overrides Overrides `json:"overrides,omitempty" yaml:"overrides,omitempty"`
44  }
```

KRM Functions Framework


```
21  func (a *ExampleApp) Schema() (*spec.Schema, error) {
22          schema, err := framework.SchemaFromFunctionDefinition(resid.NewGvk(Group, Version, Kind), CRDString)
23          return schema, errors.WrapPrefixf(err, "parsing %s schema", Kind)
24  }
```

```
49  func (a *ExampleApp) Validate() error {
50          seenDomains := make(map[string]bool)
51          for i, worker := range a.Workloads.WebWorkers {
52                  for _, domain := range worker.Domains {
53                          if seenDomains[domain] {
54                                  return errors.Errorf("duplicate domain %q in worker %d", domain, i)
55                          }
56                          seenDomains[domain] = true
57                  }
58          }
59          return nil
60  }
```

```
26  func (a *ExampleApp) Default() error {
27          if a.AppImage == "" {
28                  a.AppImage = fmt.Sprintf("registry.example.com/path/to/%s", a.ObjectMeta.Name)
29          }
```

KRM Functions Framework


```
62  func (a ExampleApp) Filter(items []*yaml.RNode) ([]*yaml.RNode, error) {
63          templates := make([]framework.ResourceTemplate, 0)
64          for _, worker := range a.Workloads.JobWorkers {
65                  templates = append(templates, framework.ResourceTemplate{
66                          Templates:    parser.TemplateFiles("templates/job_worker.template.yaml").FromFS(templateFS),
67                          TemplateData: a.jobWorkerTemplateData(worker),
68                  })
69          }
```

```
114         items, err := framework.TemplateProcessor{
115                 ResourceTemplates: templates,
116                 PatchTemplates:    patches,
117         }.Filter(items)
118         if err != nil {
119                 return nil, errors.WrapPrefixf(err, "processing templates")
120         }
```

KRM Functions Framework


```go
func processKnownAPIGroups(rl *framework.ResourceList) error {
        p := framework.VersionedAPIProcessor{FilterProvider: framework.GVKFilterMap{
                "ExampleApp": map[string]kio.Filter{
                        "platform.example.com/v1alpha1": &v1alpha1.ExampleApp{},
                },
        }}
        if err := p.Process(rl); err != nil {
                return errors.Wrap(err)
        }
        var err error
        // FormatFilter sorts the fields in a deterministic order, which makes the output
        // more suitable for review in version control.
        rl.Items, err = filters.FormatFilter{UseSchema: true}.Filter(rl.Items)
        if err != nil {
                return errors.WrapPrefixf(err, "formatting output")
        }
        return nil
}
```

KRM Functions Framework


```
app-fn path/to/example_app.yaml
```

Q&A time!

KubeCon | CloudNativeCon Europe 2023

Session resources

https://bit.ly/kubecon-1M-lines-of-yaml

Session feedback

https://sched.co/1Hycs

1M Lines of YAML:

Wrangling Kubernetes Configuration for Hundreds of Teams

KubeCon | CloudNativeCon

Europe 2023

18-21 April

Katrina Verey

Sr. Staff Software Developer

Shopify