HELM

Application deployment management for Kubernetes

About me…

- Chief of Research @codefresh.io

- github.com/alexei-led/pumba
- github.com/codefresh-io/microci

- #docker #k8s #golang #aws

- medium.com/@alexeiled

- @alexeiled

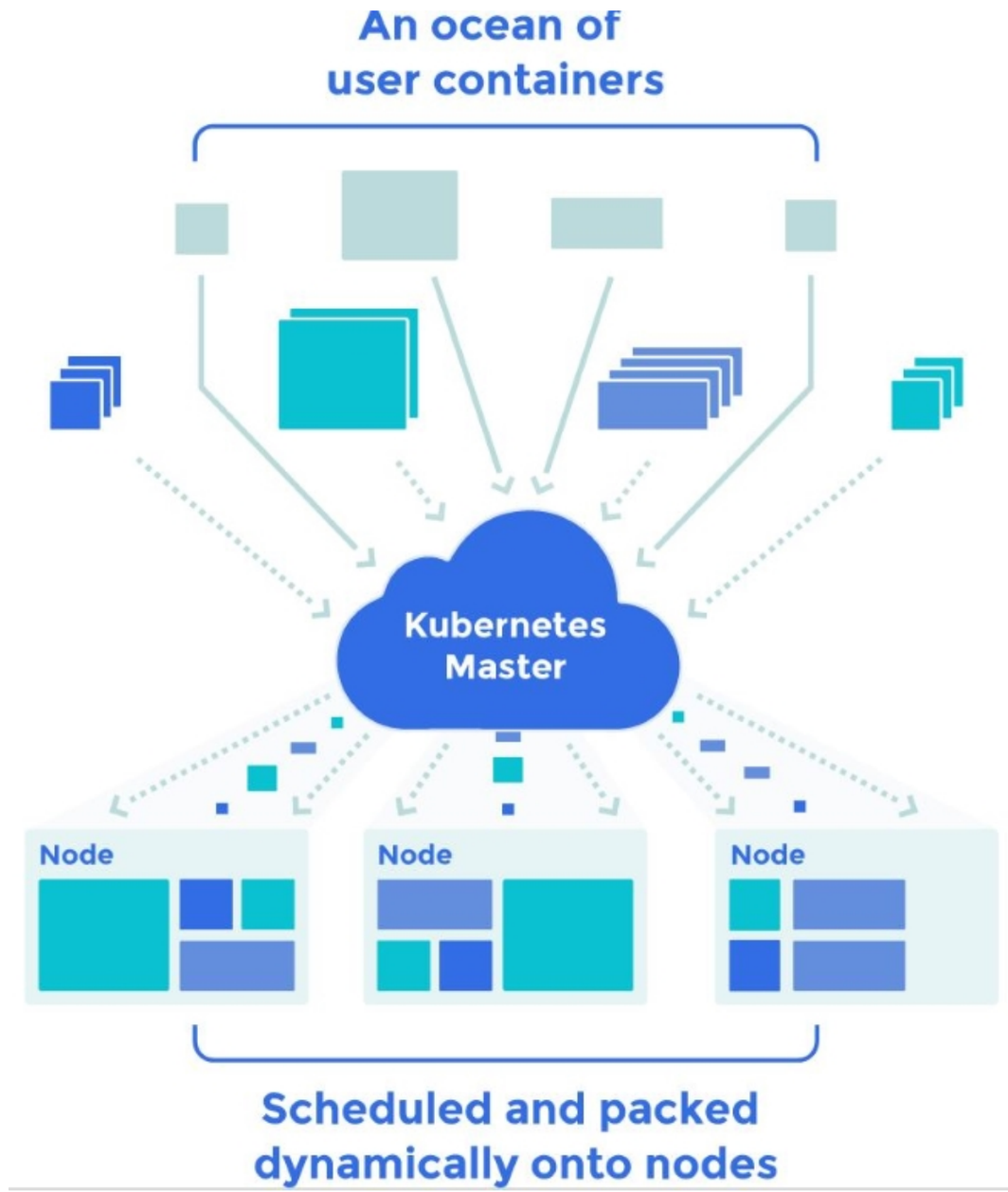
Kubernetes recap

What is Kubernetes?

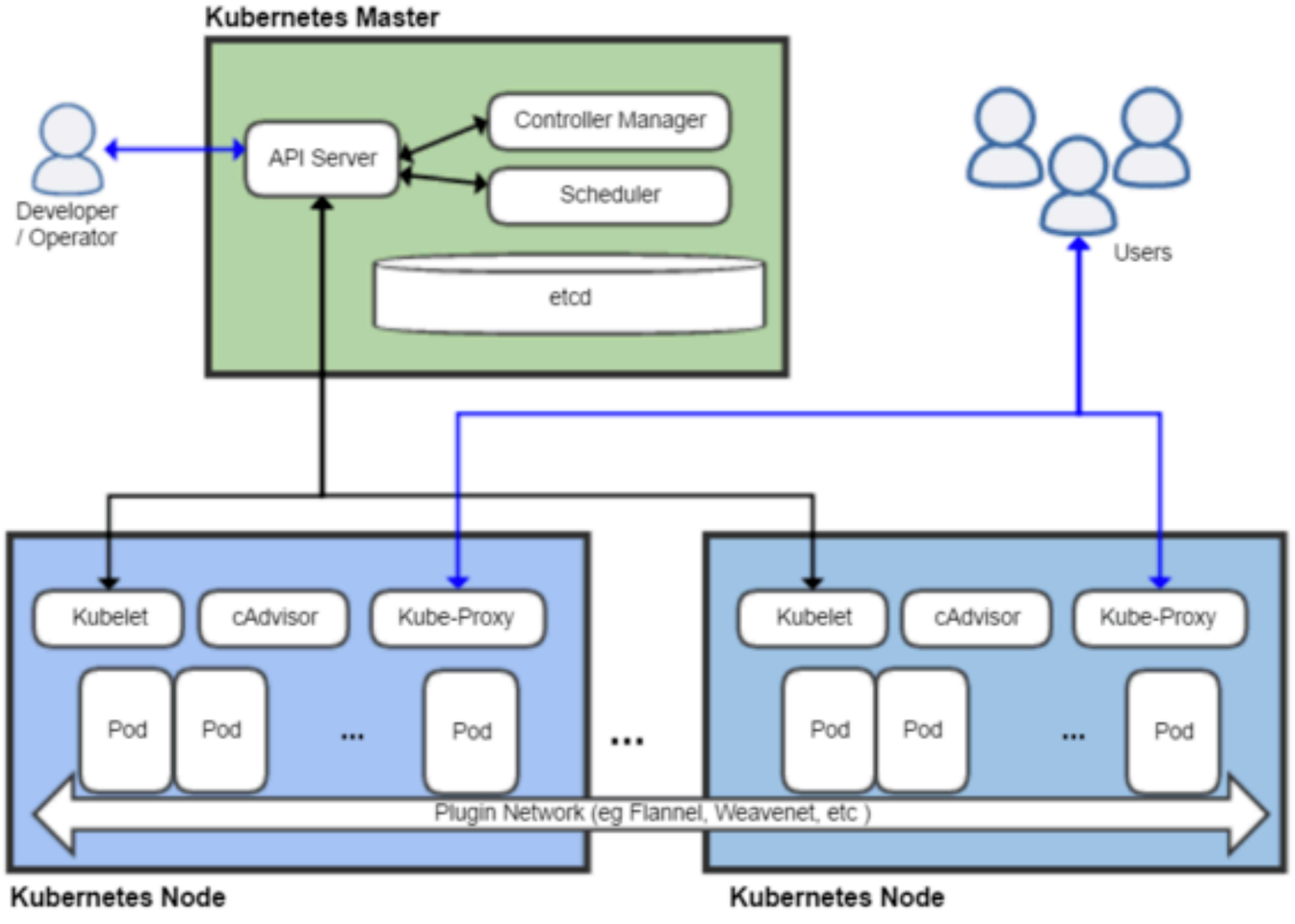
- Kubernetes is container orchestration. It’s how to run containers at scale.

- It’s a very active open-source platform with lots of contributors

- Originally developed by Google and donated to Cloud Native Computing Foundation (Linux Foundation)



K8s Architecture



kubernetes components

Julia Evans
@b0rk

★ means "required to run pods + daemonsets" -- set these up first!

● : required for some but not all API resources.

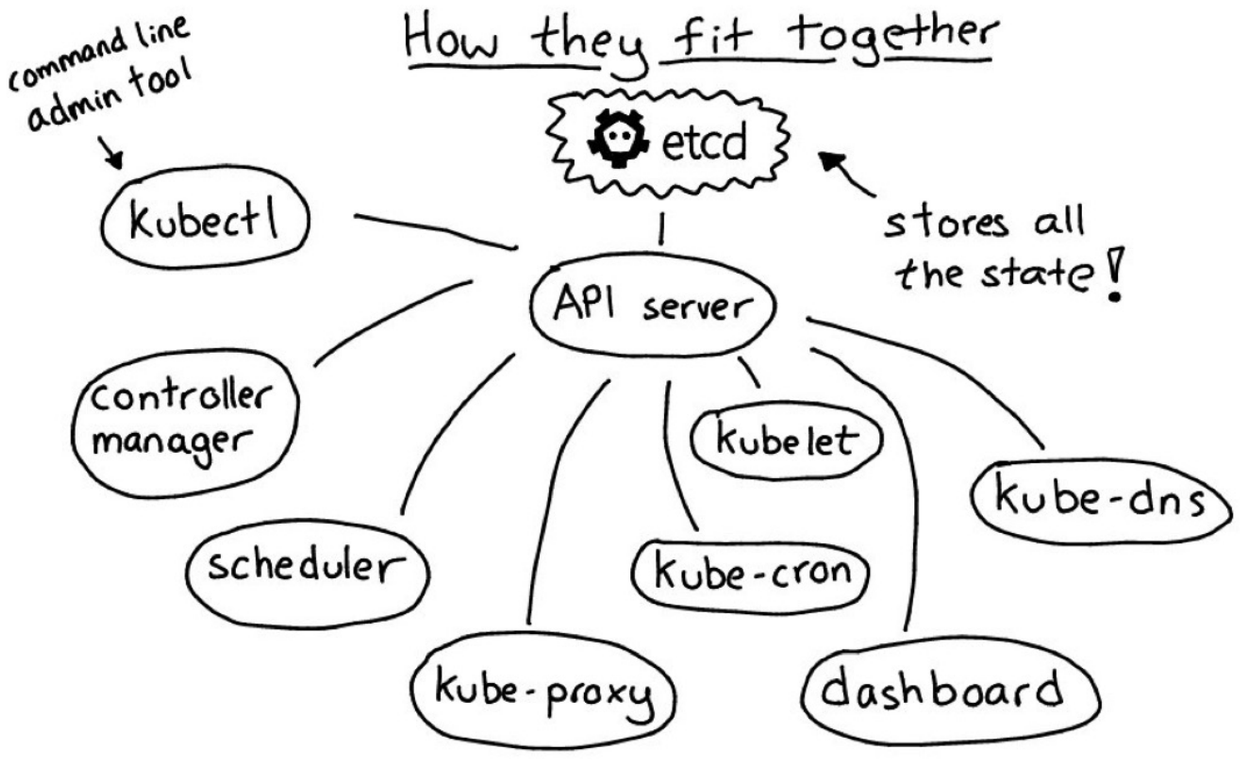
master components

★ etcd
★ API server
★ controller manager
★ scheduler
● kube-cron
(alpha, for cron jobs)
● kube-dns
(for Service DNS names
● dashboard
(a web interface!)

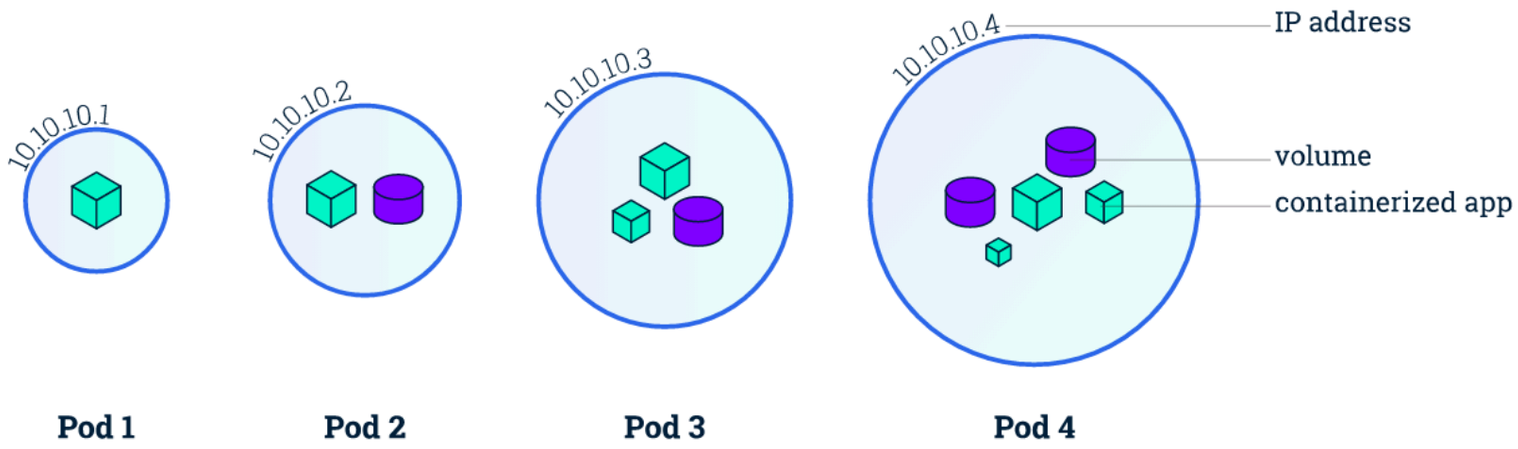
node components
(run on every node)

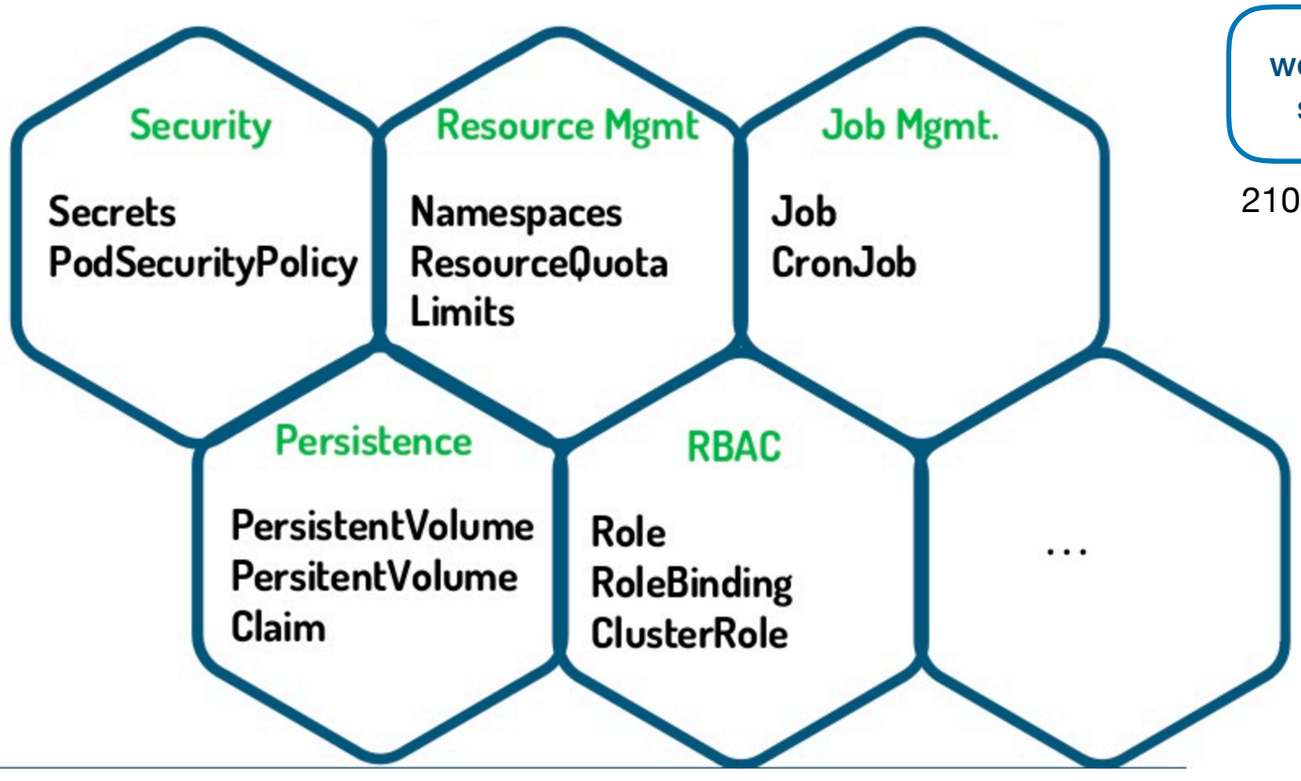
★ docker daemon
★ kubelet
(Docker + kubelet run ~ all other components)
– supervisord / systemd
(to keep kubelet + docker running)
● kube-proxy
(makes Service IPs work)

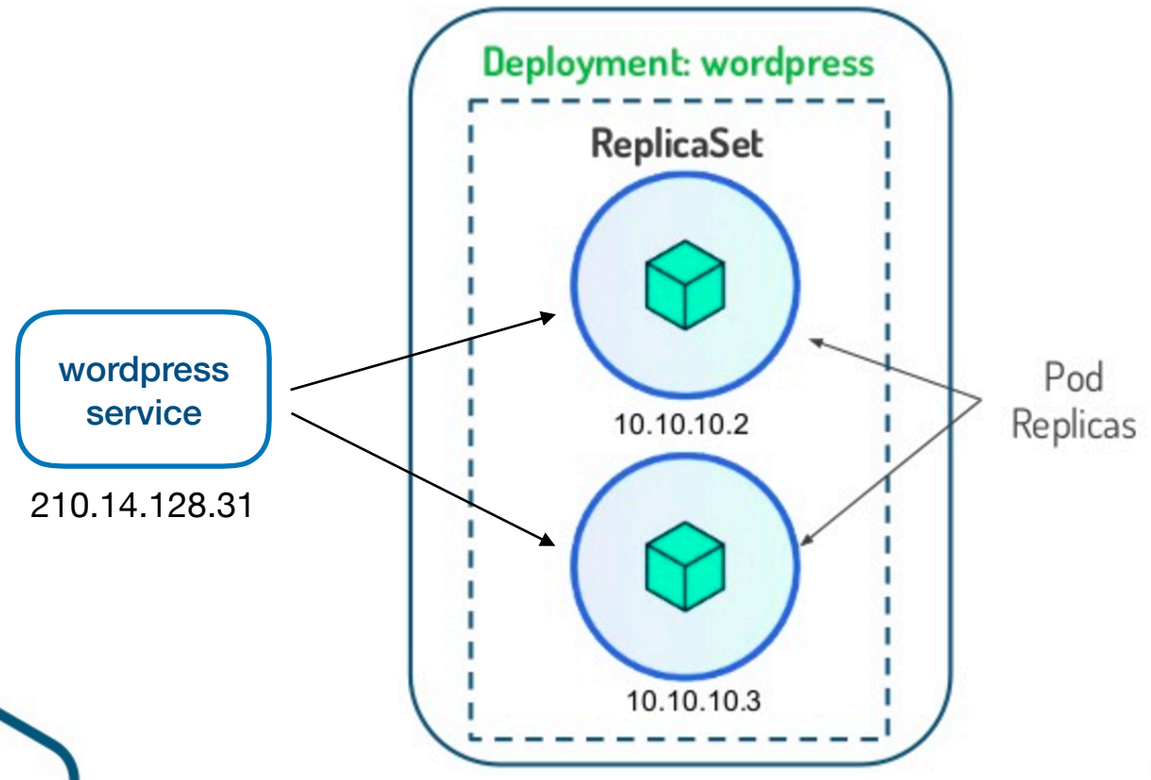
How they fit together

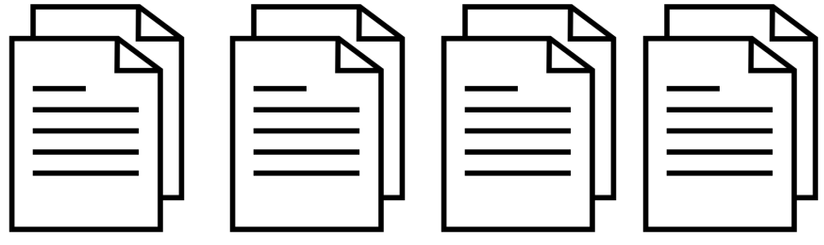


Kubernetes Objects



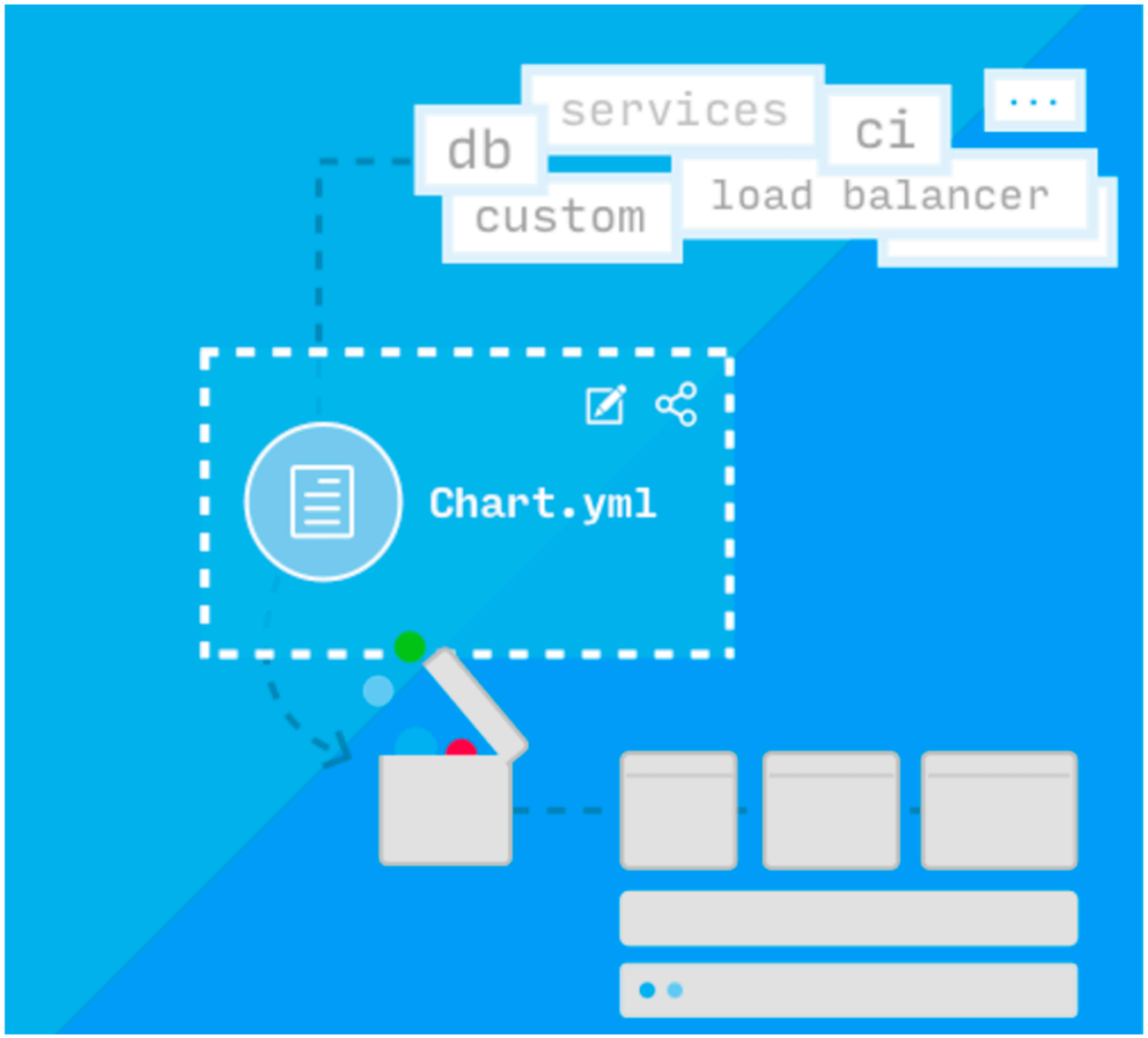






K8s Deployment Challenges

- Move to microservice architecture
    - application consists from multiple components
    - each component has its own resources and can be scaled individually
- It’s hard to …
    - … manage, edit and update multiple K8s configurations
    - … deploy multiple K8s configuration as a SINGLE application
    - … share and reuse K8s configurations and applications
    - … parametrize and support multiple environments
    - … manage application releases: rollout, rollback, diff, history
    - … define deployment lifecycle (control operations to be run in different phases)
    - … validate release state after deployment



Helm makes it **easy** to start using **Kubernetes** with **real** applications

What is Helm?

- Helm is a ***Package Manager*** for Kubernetes
  - package multiple K8s resources into a single logical deployment unit: ***Chart***
  - … but it's **not just** a *Package Manager*

- Helm is a ***Deployment Management*** for Kubernetes
  - do a repeatable deployment
  - management dependencies: reuse and share
  - manage multiple configurations
  - update, rollback and test application deployments (***Releases***)

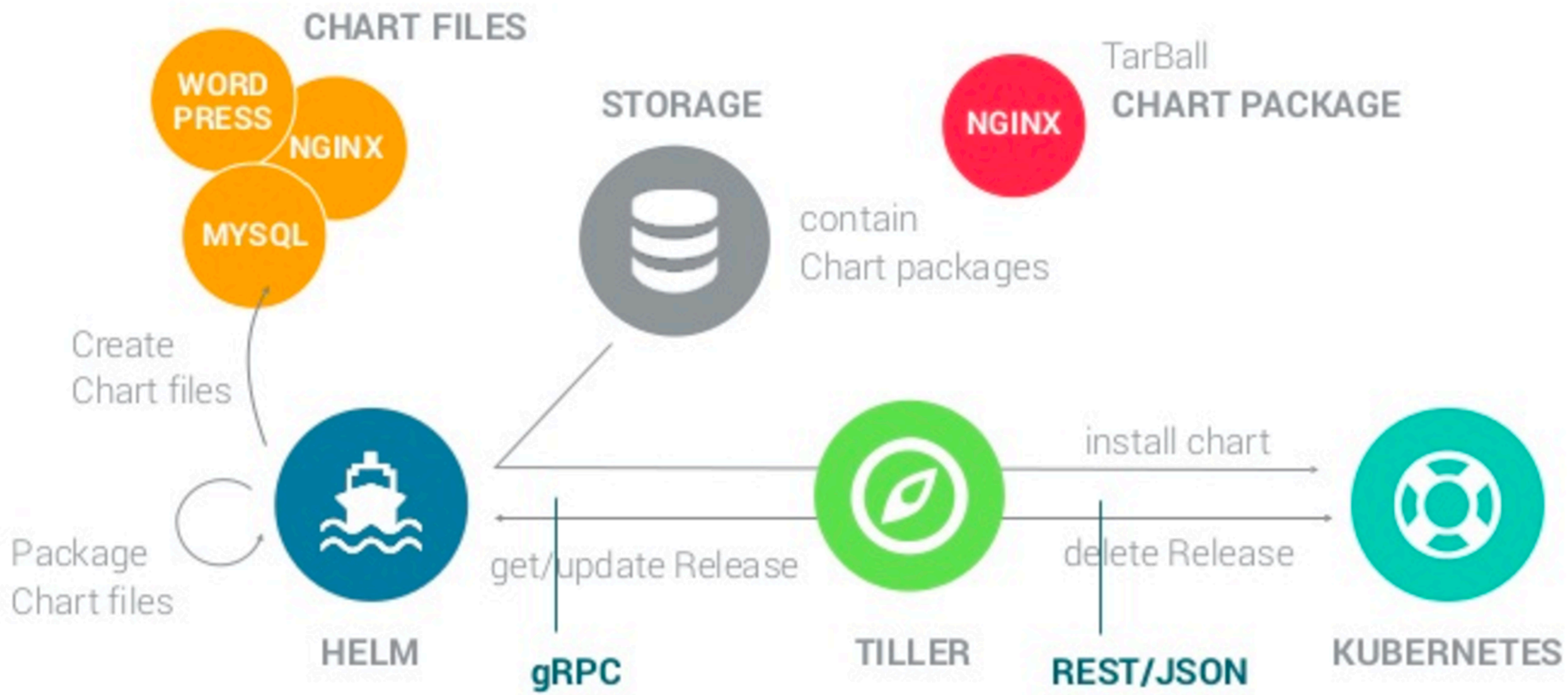
Helm Dictionary

- ***Chart*** - a package; bundle of Kubernetes resources
- ***Release*** - a chart instance is loaded into Kubernetes
  - same chart can be installed several times into the same cluster; each will have it's own *Release*
- ***Repository*** - a repository of published Charts
- ***Template*** - a K8s configuration file mixed with Go/Sprig template

kubernetes/helm

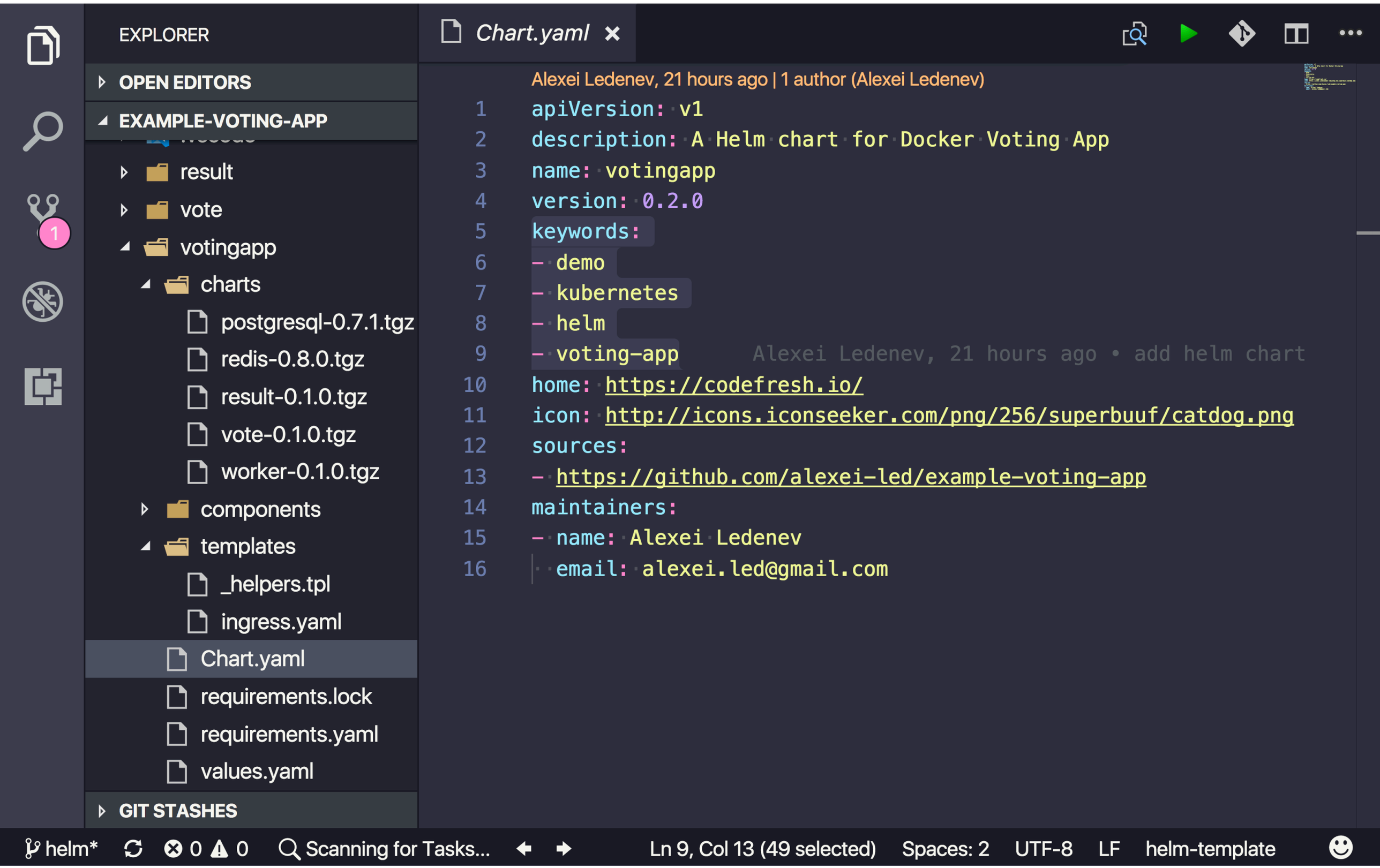
- Helm was jointly started by Google and Deis
- Helm is a Kubernetes project now (managed by CNCF)
- Active community: Google, Microsoft, Bitnami, …
- kubeapps.io - curated list (repository) of Helm Charts
- Works with any K8s cluster: K8s, Minikube, GKE, ACS, …

Helm Architecture




```
$ helm init
```

Chart Project Structure



Using Helm

```
# install helm client
brew install  kubernetes-helm
```

Dependency Management

- Helm subcharts
- `requirements.yaml`
- `helm dependency --help`

Templates

- The Go Template language: `{{.foo | quote}}`
- Variables, simple control structures (looping, conditionals, nesting)
- Pipelines - chain together templates functions
- 50+ functions from Go/Sprig Template libraries
    - date, string, conversions, encoding, reflection, data structures (list, dict), math, crypto, semver

Values

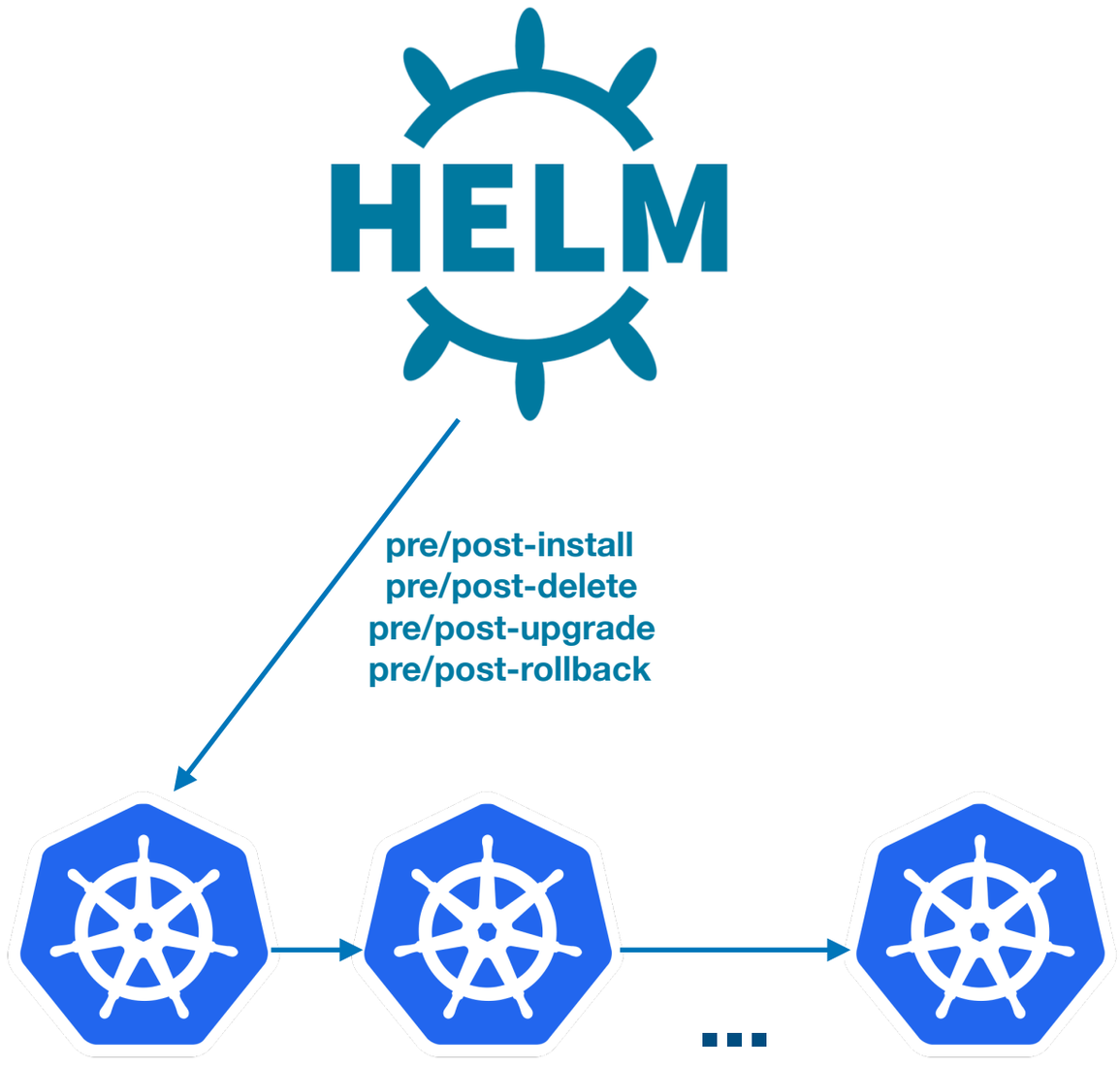
- Specify values that should be injected into templates
- Simple YAML with “namespaces”
- Each subchart can have its own `values.yaml` file
- Can use multiple Values files
- Can override individual value for install/update

Helm Hooks

- Perform "operation" at specific point of release lifecycle
- "Operation" can be any K8s resource: job, config-map, secret, pod, ...
    - The resources that a hook creates are not tracked or managed as part of the release

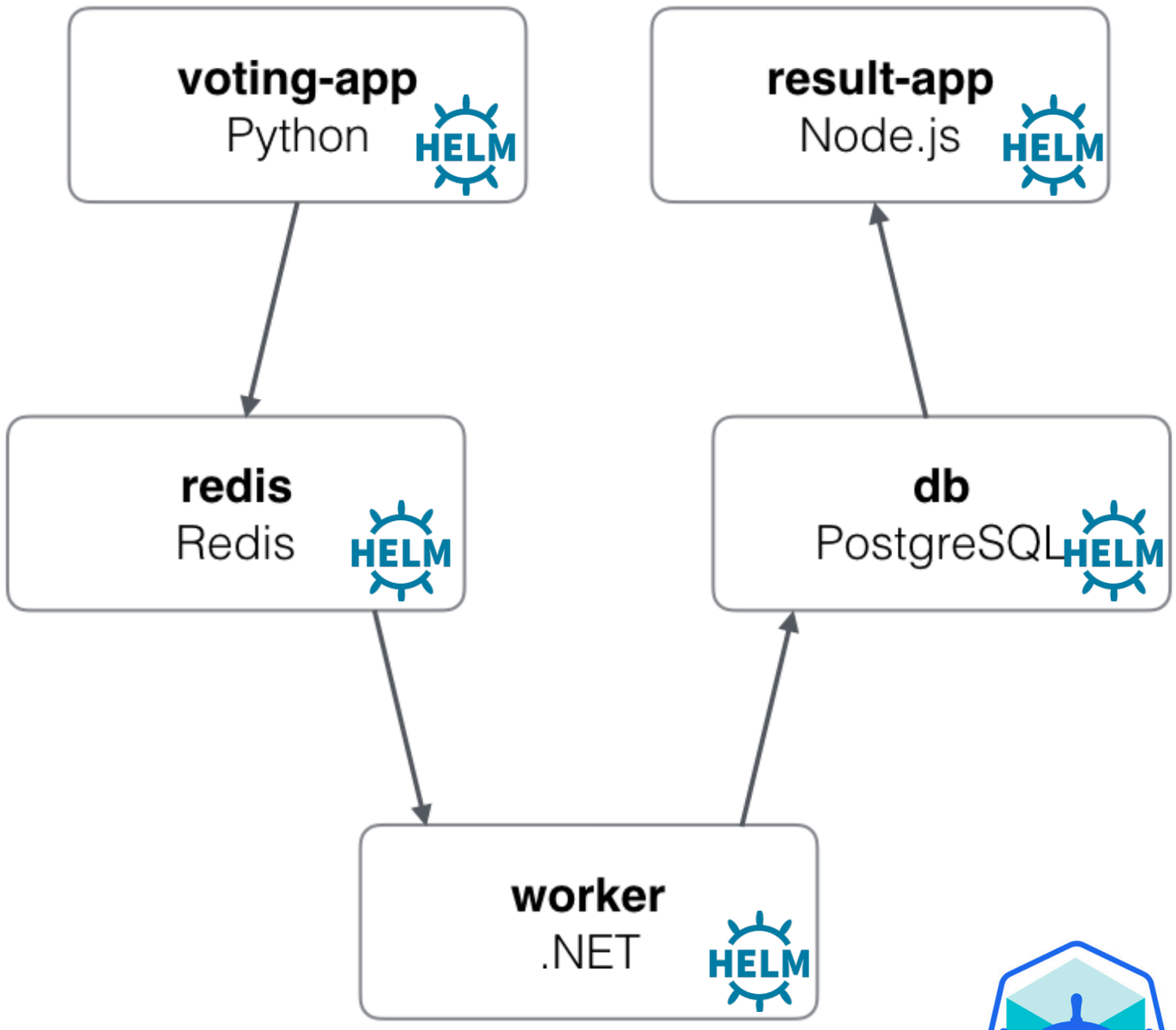
```
annotations:
  "helm.sh/hook": post-install,post-upgrade
```

```
annotations:
  "helm.sh/hook-weight": "5"
```



Demo: Voting App

https://github.com/alexei-led/example-voting-app/tree/helm



minikube

Helm Tips

1. Create ***Chart*** for each (micro)service; keep it in same Git repository

2. Learn and practice Go Template language (and Sprig template library)

3. Use Helm **hooks** to control release flow

4. Use **`helm test`** to validate releases

5. Host your own Helm **repository** for private charts; just serve `index.html` and packaged charts (can be hosted on AWS S3, Google Storage, GH pages, or other web server)

6. Manage environments with multiple Values files

7. (!) Do not commit **secrets** into GitHub; or encrypt secrets with ***sops*** or similar tool

8. Follow community Helm best practices and conventions: take a look at docs and kubernetes/charts examples

9. Use helm template plugin to debug Helm Charts; or use `--dry-run` flag

Helm Weaknesses

- Project relative immaturity - it's a young project
- No built-in environment support
- Non informative logs on failures (it's also a K8s issue)
- Weak linter: too many errors skip linter checks
- Open Issues:
  - *subcharts* are ignored when there is a *requirements.yaml*
  - the *--recreate-pods* tag is does not recreate pods of *statefulsets*

codefresh