kubernetes CONTRIBUTOR SUMMIT SAN DIEGO 2019

The KEP Implementation Journey

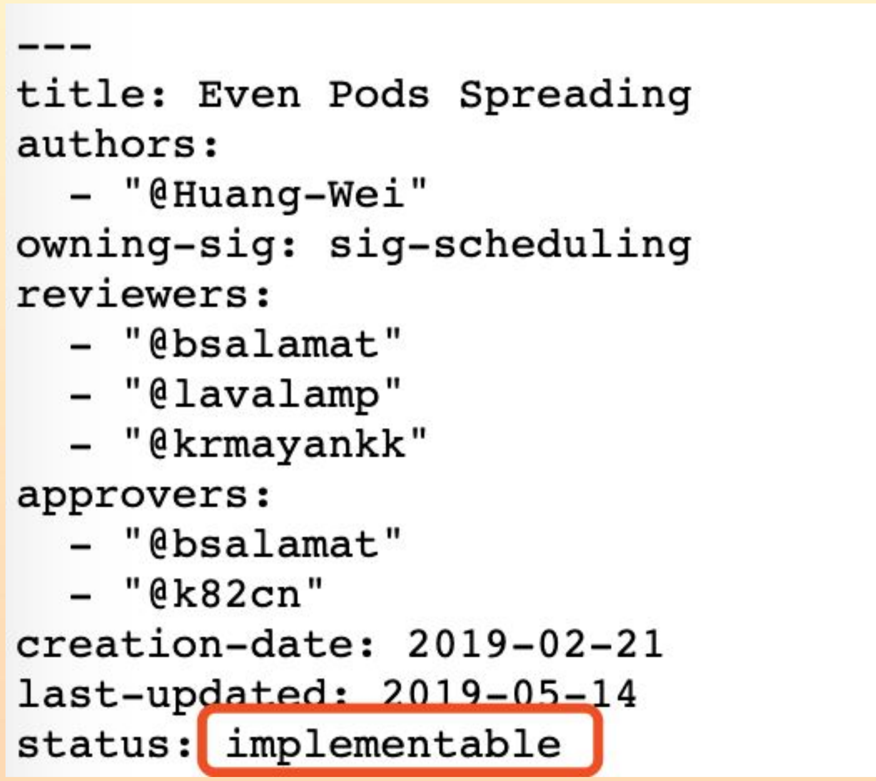
Wei Huang, Senior Software Engineer, IBM

KEP Basics

- KEP: Kubernetes Enhancement Proposal
  - Consistent feature lifecycle management
  - Serve many audiences
  - https://github.com/kubernetes/enhancements/tree/master/keps

Before Starting

- **Prerequisite**
  - A thoroughly-discussed KEP
  - KEP tagged as “implementable”
  - Negotiate with SIG(s) leaders about a rough roadmap and align with release schedule
  - Reserve your bandwidth



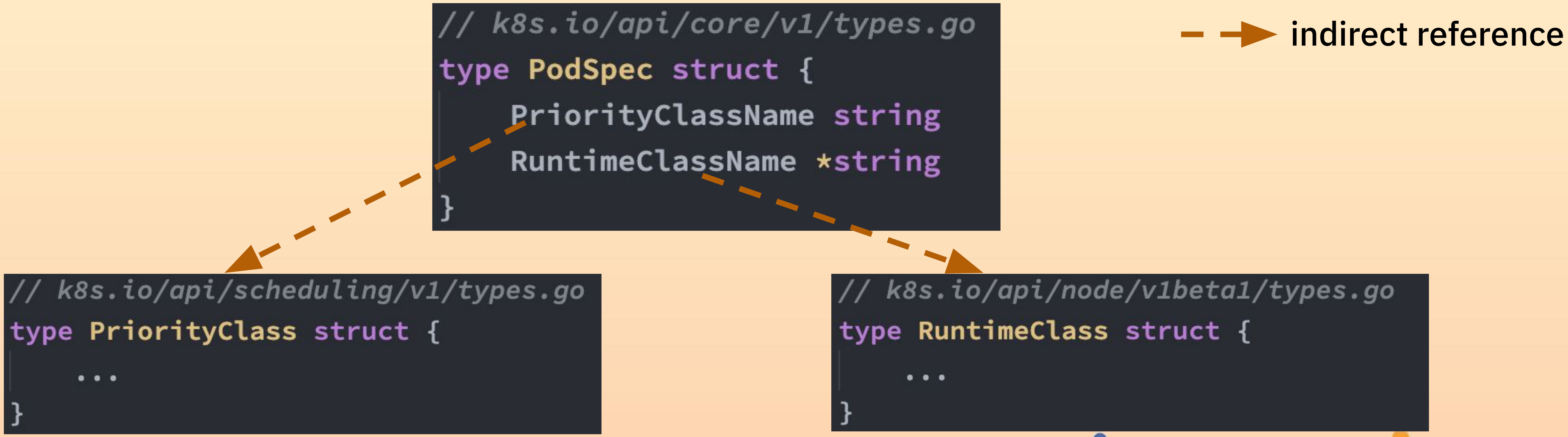
```
---
title: Even Pods Spreading
authors:
  - "@Huang-Wei"
owning-sig: sig-scheduling
reviewers:
  - "@bsalamat"
  - "@lavalamp"
  - "@krmayankk"
approvers:
  - "@bsalamat"
  - "@k82cn"
creation-date: 2019-02-21
last-updated: 2019-05-14
status: implementable
```

Scope, Break Down and Prioritize Items

- **Ask yourself some questions**
  - Is new API needed? If so, what kind of API?
  - In which component(s) to implement? (sometimes the answer is not that obvious, e.g. TaintBasedEviction)
  - Update KEP (esp. API change) if necessary
- 🎉 **Output: an umbrella issue to ensure everyone on the same page**
  - EvenPodsSpread (solo) - #77284
  - Scheduler Framework (cooperation) - #83554

API Implementation

- API design pattern
  - Top-level API and connect to other API (RuntimeClass, Priority, etc.)



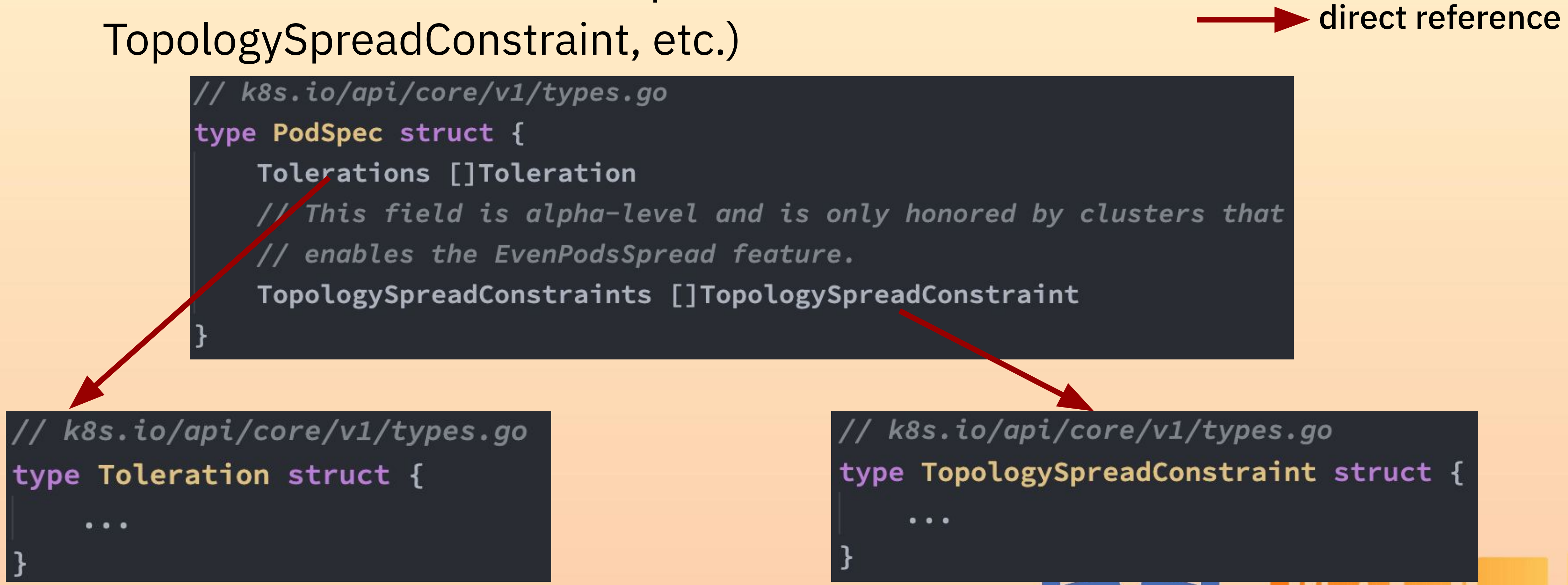
```go
// k8s.io/api/core/v1/types.go
type PodSpec struct {
    PriorityClassName string
    RuntimeClassName *string
}
```

```go
// k8s.io/api/scheduling/v1/types.go
type PriorityClass struct {
    ...
}
```

```go
// k8s.io/api/node/v1beta1/types.go
type RuntimeClass struct {
    ...
}
```

API Implementation

- API design pattern (cont.)
  - Sublevel-API attached to top-level API (Toleration, TopologySpreadConstraint, etc.)



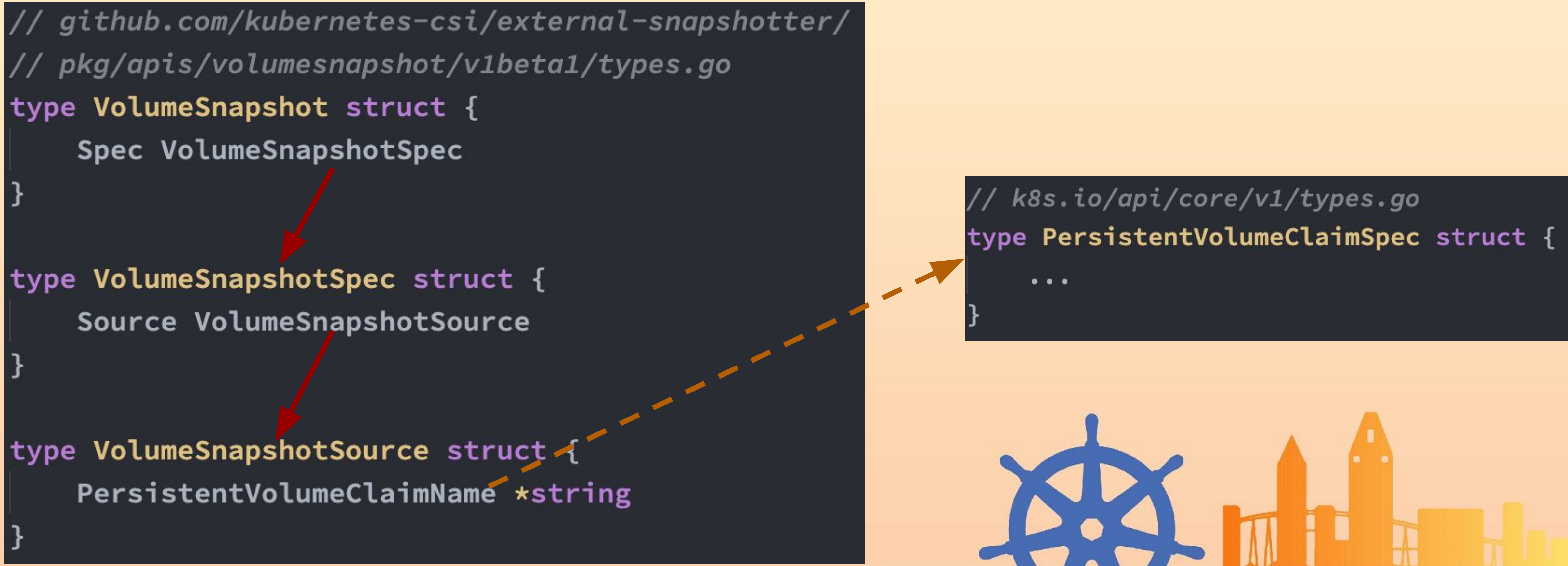
```go
// k8s.io/api/core/v1/types.go
type PodSpec struct {
    Tolerations []Toleration
    // This field is alpha-level and is only honored by clusters that
    // enables the EvenPodsSpread feature.
    TopologySpreadConstraints []TopologySpreadConstraint
}
```

```go
// k8s.io/api/core/v1/types.go
type Toleration struct {
    ...
}
```

```go
// k8s.io/api/core/v1/types.go
type TopologySpreadConstraint struct {
    ...
}
```

API Implementation

- API design pattern (cont.)
  - CRD (VolumeSnapshot, PodGroup, etc.)

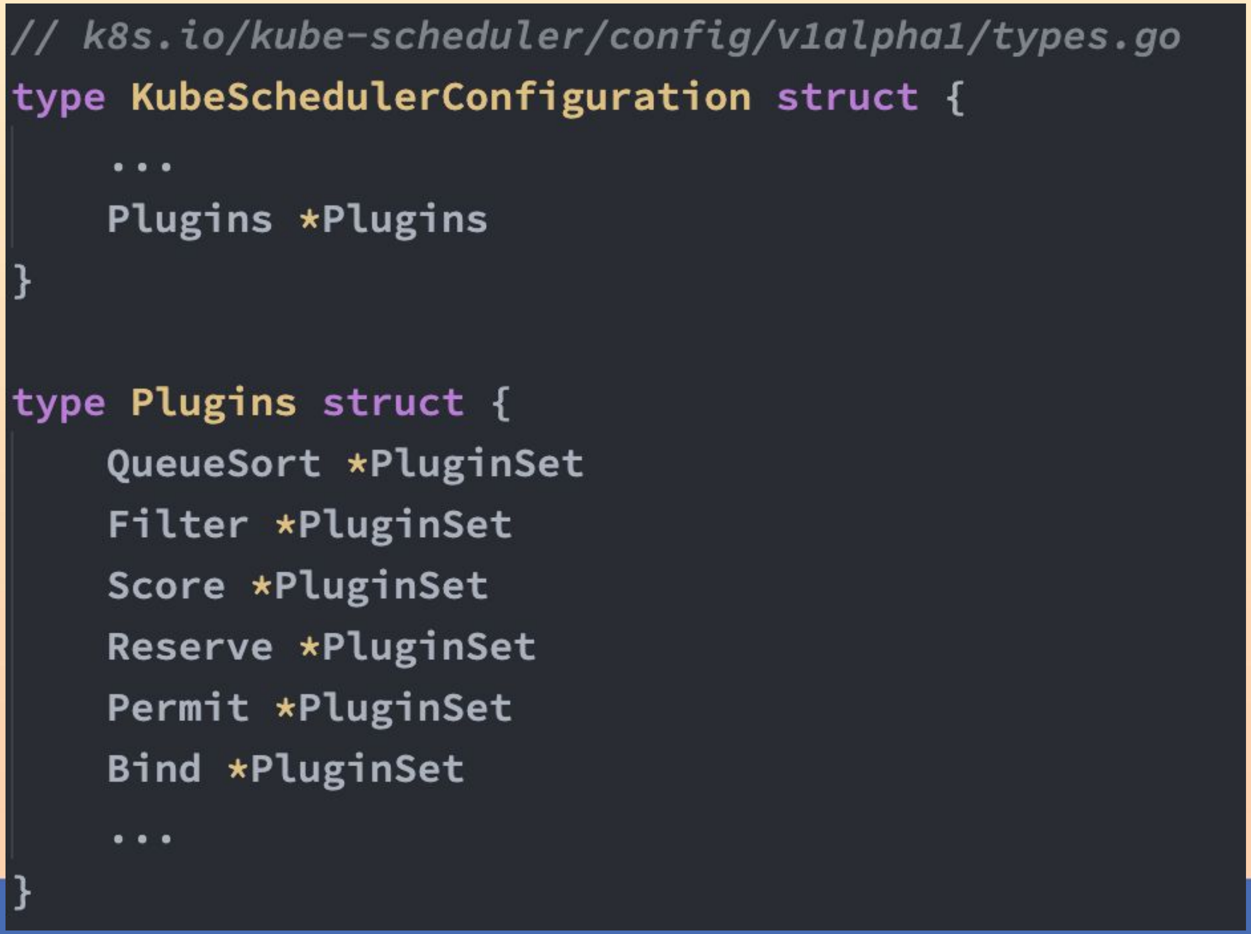


```go
// github.com/kubernetes-csi/external-snapshotter/
// pkg/apis/volumesnapshot/v1beta1/types.go
type VolumeSnapshot struct {
    Spec VolumeSnapshotSpec
}

type VolumeSnapshotSpec struct {
    Source VolumeSnapshotSource
}

type VolumeSnapshotSource struct {
    PersistentVolumeClaimName *string
}
```

```go
// k8s.io/api/core/v1/types.go
type PersistentVolumeClaimSpec struct {
    ...
}
```

API Implementation

- API design pattern (cont.)
  - ComponentConfig (Scheduler Framework, etc.)

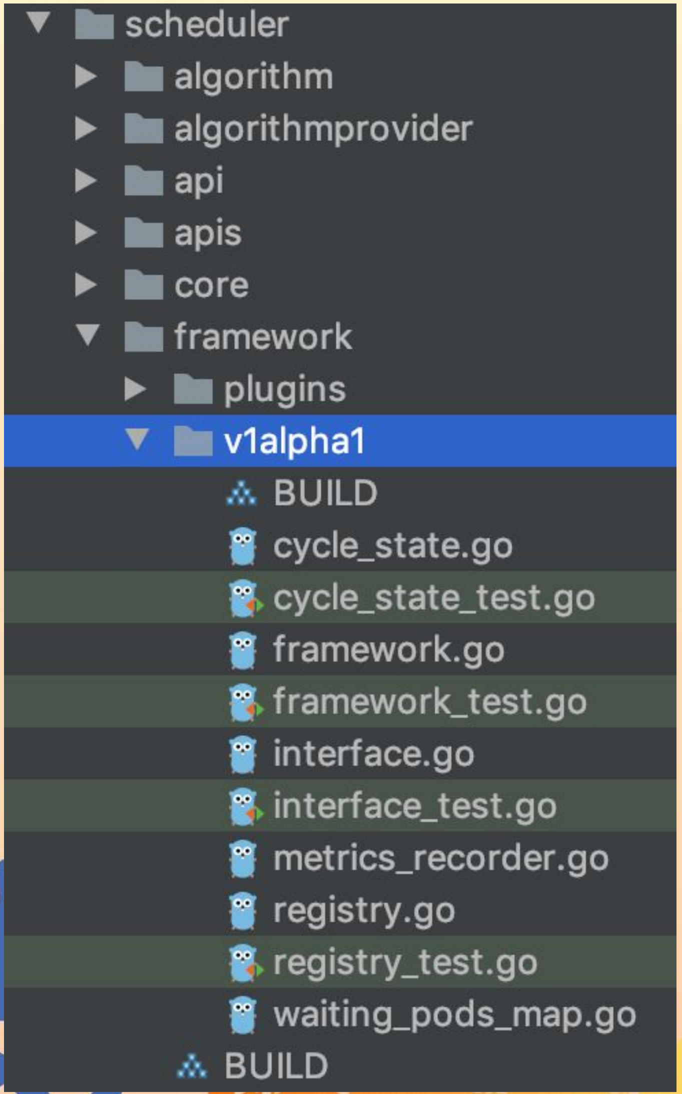


```go
// k8s.io/kube-scheduler/config/v1alpha1/types.go
type KubeSchedulerConfiguration struct {
    ...
    Plugins *Plugins
}

type Plugins struct {
    QueueSort *PluginSet
    Filter *PluginSet
    Score *PluginSet
    Reserve *PluginSet
    Permit *PluginSet
    Bind *PluginSet
    ...
}
```

kubernetes CONTRIBUTOR SUMMIT SAN DIEGO 2019

API Implementation

- **API design pattern (cont.)**
  - Component internal API
    - Scheduler Framework - pkg/scheduler/framework/v1alpha1
    - Part of Scheduler Extender - pkg/scheduler/apis/extender/v1
    - Kubelet
      - pkg/kubelet/apis/podresources/v1alpha1 - protobuf style
      - pkg/kubelet/apis/stats/v1alpha1
  - Usually involves limited or none codegens
  - May not need data persistence



```
▼ scheduler
  ▶ algorithm
  ▶ algorithmprovider
  ▶ api
  ▶ apis
  ▶ core
  ▼ framework
    ▶ plugins
    ▼ v1alpha1
        BUILD
        cycle_state.go
        cycle_state_test.go
        framework.go
        framework_test.go
        interface.go
        interface_test.go
        metrics_recorder.go
        registry.go
        registry_test.go
        waiting_pods_map.go
      BUILD
```

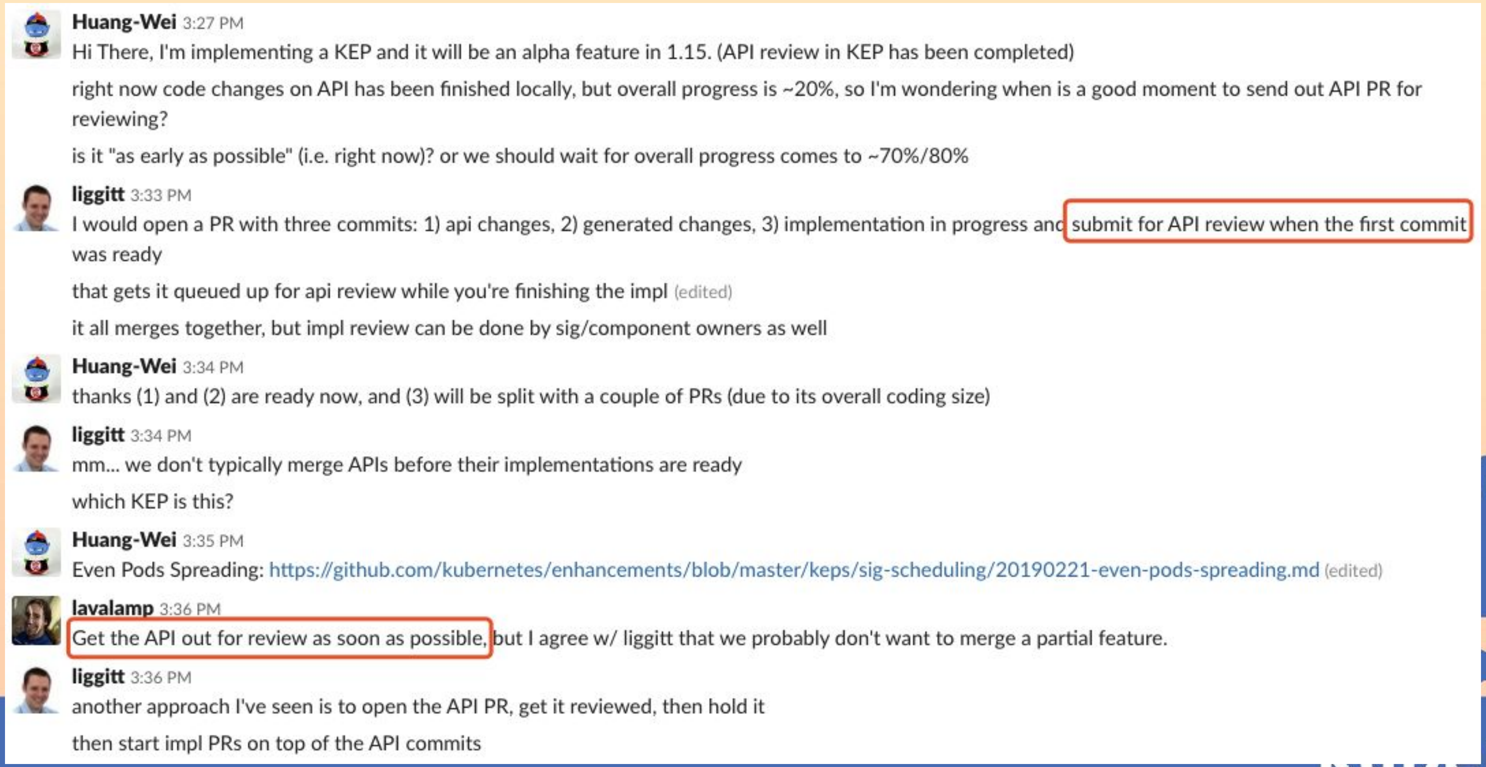
API Implementation

- 🎉 **Output: an API PR (example)**
  - New spec - internal or (and) external
  - GVK (group, version, kind) definition
  - Code Generation
  - Validation, Defaulting
- **Codegen commands (works in 1.16)**

```
   make generated_files
   hack/update-generated-protobuf.sh
   hack/update-codegen.sh
   UPDATE_COMPATIBILITY_FIXTURE_DATA=true go test
./vendor/k8s.io/api -run //HEAD &>/dev/null

   hack/update-generated-swagger-docs.sh
   hack/update-openapi-spec.sh
```

API Implementation

- Tips
  - contributors/devel/sig-architecture/api-conventions.md
  - contributors/devel/sig-architecture/api_changes.md
  - Request API code review as early as possible



Huang-Wei 3:27 PM
Hi There, I'm implementing a KEP and it will be an alpha feature in 1.15. (API review in KEP has been completed)
right now code changes on API has been finished locally, but overall progress is ~20%, so I'm wondering when is a good moment to send out API PR for reviewing?
is it "as early as possible" (i.e. right now)? or we should wait for overall progress comes to ~70%/80%

liggitt 3:33 PM
I would open a PR with three commits: 1) api changes, 2) generated changes, 3) implementation in progress and submit for API review when the first commit was ready
that gets it queued up for api review while you're finishing the impl (edited)
it all merges together, but impl review can be done by sig/component owners as well

Huang-Wei 3:34 PM
thanks (1) and (2) are ready now, and (3) will be split with a couple of PRs (due to its overall coding size)

liggitt 3:34 PM
mm... we don't typically merge APIs before their implementations are ready
which KEP is this?

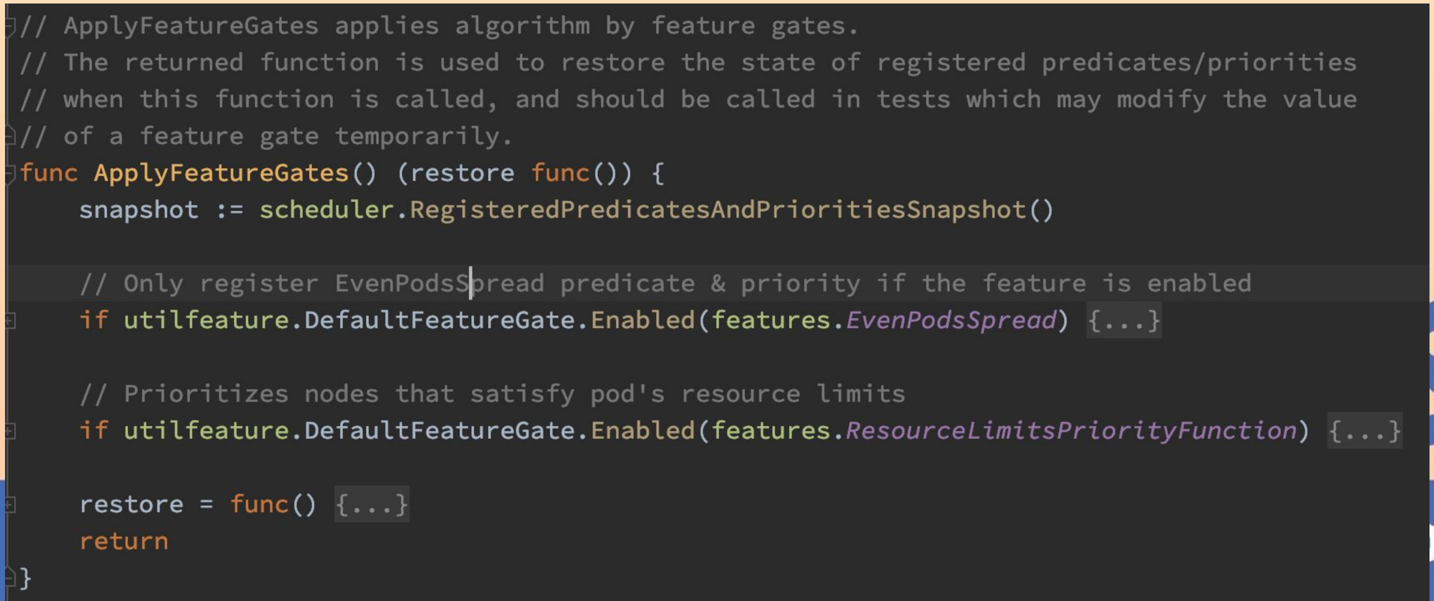
Huang-Wei 3:35 PM
Even Pods Spreading: https://github.com/kubernetes/enhancements/blob/master/keps/sig-scheduling/20190221-even-pods-spreading.md (edited)

lavalamp 3:36 PM
Get the API out for review as soon as possible, but I agree w/ liggitt that we probably don't want to merge a partial feature.

liggitt 3:36 PM
another approach I've seen is to open the API PR, get it reviewed, then hold it
then start impl PRs on top of the API commits

Core Logic Implementation

- Learn from existing code
- If across multiple components, define the responsibility for each
- Users not using the new feature shouldn't be punished
- Feature gating
  - global feature gate - control core kube components
  - internal feature gating logic - run feature gate check just once



```go
// ApplyFeatureGates applies algorithm by feature gates.
// The returned function is used to restore the state of registered predicates/priorities
// when this function is called, and should be called in tests which may modify the value
// of a feature gate temporarily.
func ApplyFeatureGates() (restore func()) {
    snapshot := scheduler.RegisteredPredicatesAndPrioritiesSnapshot()

    // Only register EvenPodsSpread predicate & priority if the feature is enabled
    if utilfeature.DefaultFeatureGate.Enabled(features.EvenPodsSpread) {...}

    // Prioritizes nodes that satisfy pod's resource limits
    if utilfeature.DefaultFeatureGate.Enabled(features.ResourceLimitsPriorityFunction) {...}

    restore = func() {...}
    return
}
```

Core Logic Implementation

- **Implement each sub-feature, with tests covered**
- **Managing PR dependencies and git branches**
  - Each PR serves as a standalone functional unit
  - Understand PR dependencies (#77284, #83554) for better cooperation
  - (optional) Create a separate feature branch on k/k
  - Keep coding, don't stop to await on-going code reviews

Core Logic Implementation

- **Managing PR dependencies and git branches (cont.)**
  - Develop PRs <u>in parallel</u> to leverage CI
    - **PR1** - commits **A, B**
    - **PR2** - branched off **PR1**, commits **A, B, C**
      - rebase after **PR1** gets merged, so **PR2** only shows commit **C**
    - **PR3** - branched off **PR2**, commits **A, B, C, D, E**
      - rebase after **PR2** gets merged, so **PR3** only shows commits **D, E**
- 🎉 **Output: a series of PRs implementing core logic, and sufficient tests**
  - Get along well with your friend `git rebase` :)

Test Strategies

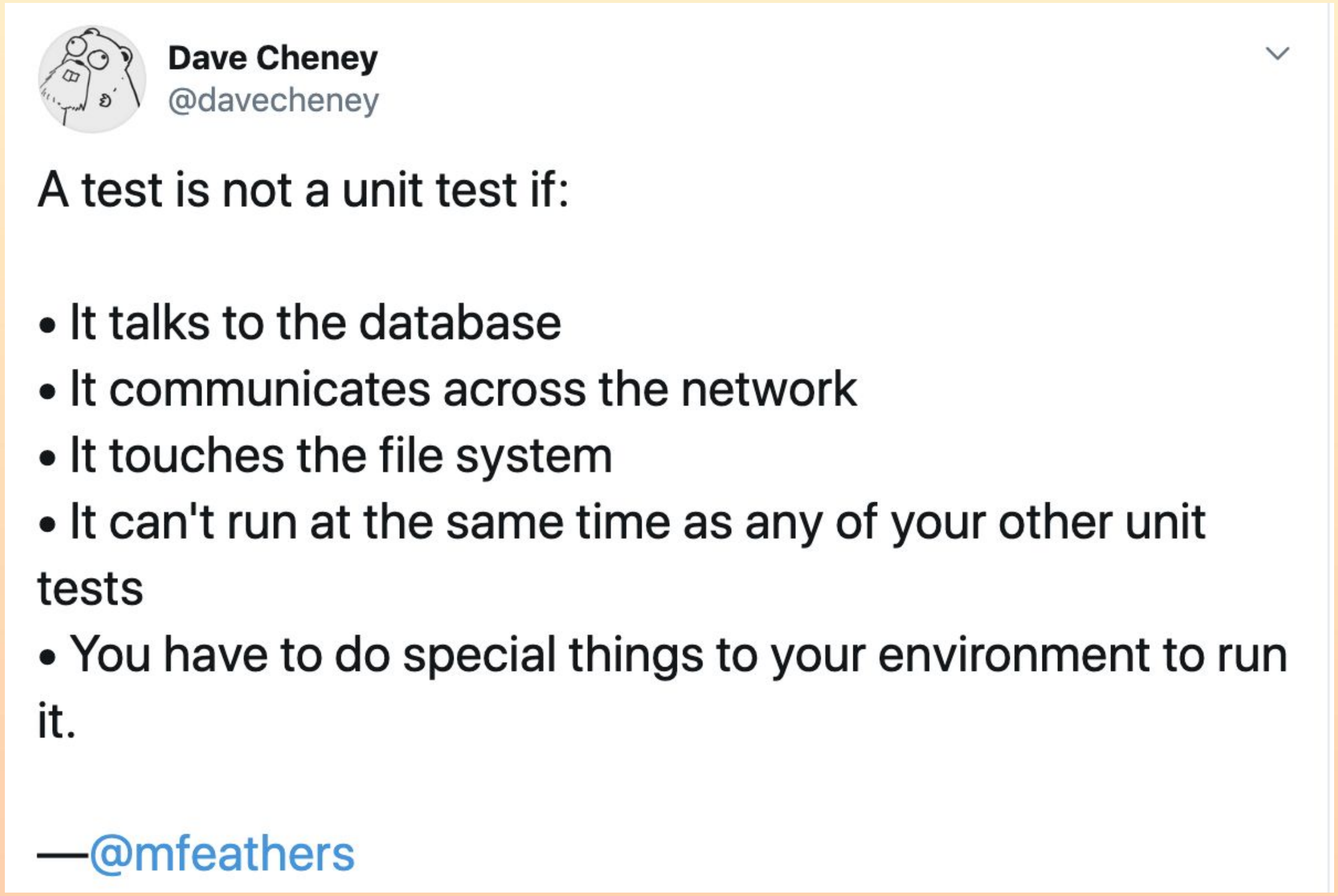
- Unit Tests
  - **must** be equipped with each PR
    - mandatory for public functions
    - good-to-have for critical private functions
  - before `git push`, ensure local UT passes
    - `go test k8s.io/kubernetes/pkg/scheduler/...`
  - detect racing condition
    - `go test <pkg> --race`
    - `go test <pkg> --race --count=<number>`
  - follow the Golang "TableDrivenTest" convention
  - disable caching
    - `go test <pkg> --count=1`

Test Strategies

- **Unit Tests (cont.)**
  - avoid flakes
    - `go test <pkg> --count=<big number>`
    - be aware of shared variable (featuregate, env variables, tmpdir, etc.)
    - be aware of goroutine leaks
  - unable to compose UTs usually implies *smelly* code
  - concrete struct vs. interface
  - mocking examples
    - FakeClientset, FakeAPIObj (core API, CRD, ... auto-generated)
    - FakeClock
    - Test executable - Real Binary vs. Mock
    - Test CMD options

Test Strategies

- Unit Tests (cont.)



Dave Cheney
@davecheney

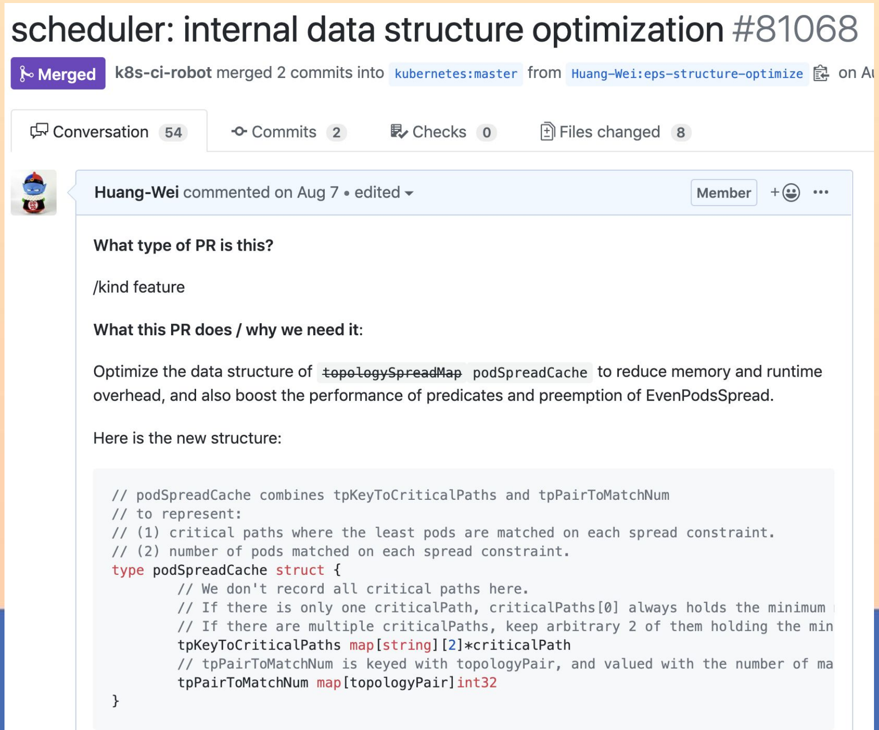
A test is not a unit test if:

• It talks to the database
• It communicates across the network
• It touches the file system
• It can't run at the same time as any of your other unit tests
• You have to do special things to your environment to run it.

—@mfeathers

Test Strategies

- Benchmark Test
  - focus on performance instead of functionalities
  - leverage Golang benchmark testing facilities (`*testing.B`)
    - `go test --run=^$ <pkg> --bench .` (replace `.` with `^<func name>$`)



scheduler: internal data structure optimization #81068

Merged k8s-ci-robot merged 2 commits into `kubernetes:master` from `Huang-Wei:eps-structure-optimize` on Au

Conversation 54 · Commits 2 · Checks 0 · Files changed 8

Huang-Wei commented on Aug 7 • edited · Member

What type of PR is this?

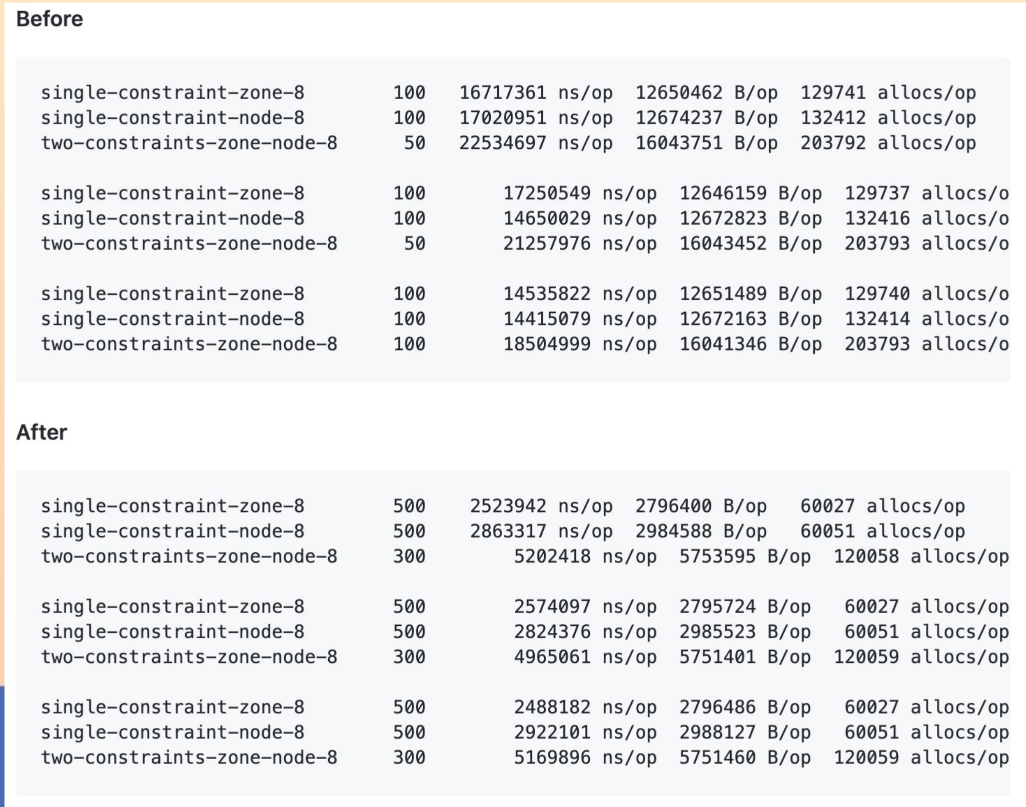
/kind feature

What this PR does / why we need it:

Optimize the data structure of ~~`topologySpreadMap`~~ `podSpreadCache` to reduce memory and runtime overhead, and also boost the performance of predicates and preemption of EvenPodsSpread.

Here is the new structure:

```go
// podSpreadCache combines tpKeyToCriticalPaths and tpPairToMatchNum
// to represent:
// (1) critical paths where the least pods are matched on each spread constraint.
// (2) number of pods matched on each spread constraint.
type podSpreadCache struct {
        // We don't record all critical paths here.
        // If there is only one criticalPath, criticalPaths[0] always holds the minimum
        // If there are multiple criticalPaths, keep arbitrary 2 of them holding the min
        tpKeyToCriticalPaths map[string][2]*criticalPath
        // tpPairToMatchNum is keyed with topologyPair, and valued with the number of ma
        tpPairToMatchNum map[topologyPair]int32
}
```



Before

```
single-constraint-zone-8         100   16717361 ns/op  12650462 B/op  129741 allocs/op
single-constraint-node-8         100   17020951 ns/op  12674237 B/op  132412 allocs/op
two-constraints-zone-node-8       50   22534697 ns/op  16043751 B/op  203792 allocs/op

single-constraint-zone-8         100       17250549 ns/op  12646159 B/op  129737 allocs/o
single-constraint-node-8         100       14650029 ns/op  12672823 B/op  132416 allocs/o
two-constraints-zone-node-8       50       21257976 ns/op  16043452 B/op  203793 allocs/o

single-constraint-zone-8         100       14535822 ns/op  12651489 B/op  129740 allocs/o
single-constraint-node-8         100       14415079 ns/op  12672163 B/op  132414 allocs/o
two-constraints-zone-node-8      100       18504999 ns/op  16041346 B/op  203793 allocs/o
```

After

```
single-constraint-zone-8         500    2523942 ns/op  2796400 B/op   60027 allocs/op
single-constraint-node-8         500    2863317 ns/op  2984588 B/op   60051 allocs/op
two-constraints-zone-node-8      300       5202418 ns/op  5753595 B/op  120058 allocs/op

single-constraint-zone-8         500       2574097 ns/op  2795724 B/op   60027 allocs/op
single-constraint-node-8         500       2824376 ns/op  2985523 B/op   60051 allocs/op
two-constraints-zone-node-8      300       4965061 ns/op  5751401 B/op  120059 allocs/op

single-constraint-zone-8         500       2488182 ns/op  2796486 B/op   60027 allocs/op
single-constraint-node-8         500       2922101 ns/op  2988127 B/op   60051 allocs/op
two-constraints-zone-node-8      300       5169896 ns/op  5751460 B/op  120059 allocs/op
```

Test Strategies

- Benchmark Test - from Unit Test to Benchmark Test

```go
func TestFoo(t *testing.T) {
    tests := []struct {
        name    string
        args    ...
        want    retType
    } {
        {
            name: "<description for case1>"
            ...
        }
    }

    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            // build dependencies
            r := buildR()
            got := r.Foo(tt.args)
            if !reflect.DeepEqual(got, tt.want) {
                t.Errorf("foo() = %v, want %v", got, tt.want)
            }
        })
    }
}
```

t *testing.t => **b *testing.B**

```go
func BenchmarkFoo(b *testing.B) {
    tests := []struct {
        name    string
        args    ...
    } {
        {
            name: "<description for case1>"
            ...
        }
    }

    for _, tt := range tests {
        b.Run(tt.name, func(b *testing.B) {
            // build dependencies
            r := buildR()
            b.ResetTimer()
            for i := 0; i < b.N; i++ {
                // perform the operation we're analyzing
                r.Foo(tt.args)
            }
        })
    }
}
```

remove **want** field

reset timer

b.**N**

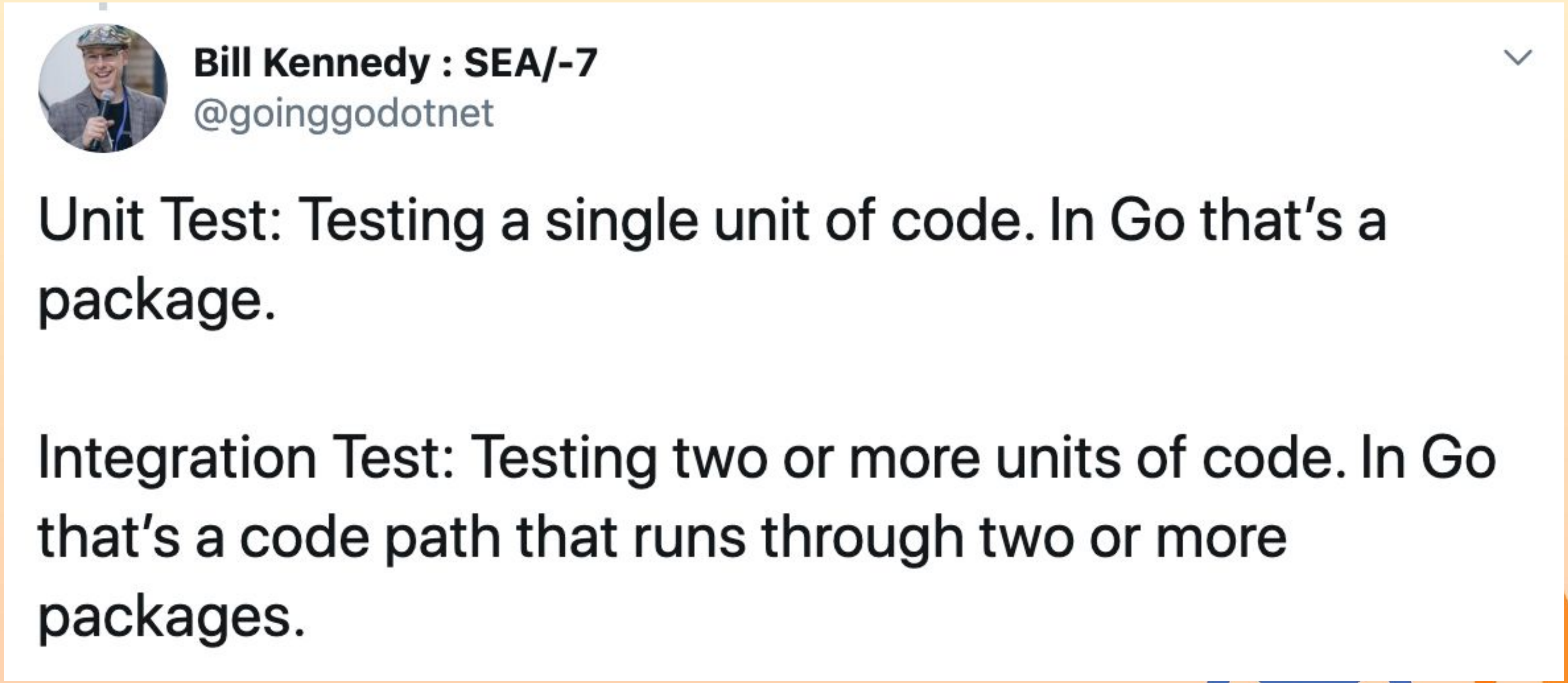
No need to verify correctness

Test Strategies

- Integration Test
  - test sub-component (package) interactions, e.g. EvenPodsSpread
    - predicates, preemption, priorities, priorities+predicates
  - test component interactions (Controller Manager + Scheduler + Kubelet + APIServer), e.g. TaintBasedEviction
    - node lifecycle manager (part of Controller Manager)
    - scheduler
    - admissions (part of APIServer)
    - some goroutines to simulate kubelet
  - usually require etcd/APIServer
  - essentially Unit Test

Test Strategies

- Integration Test (cont.)



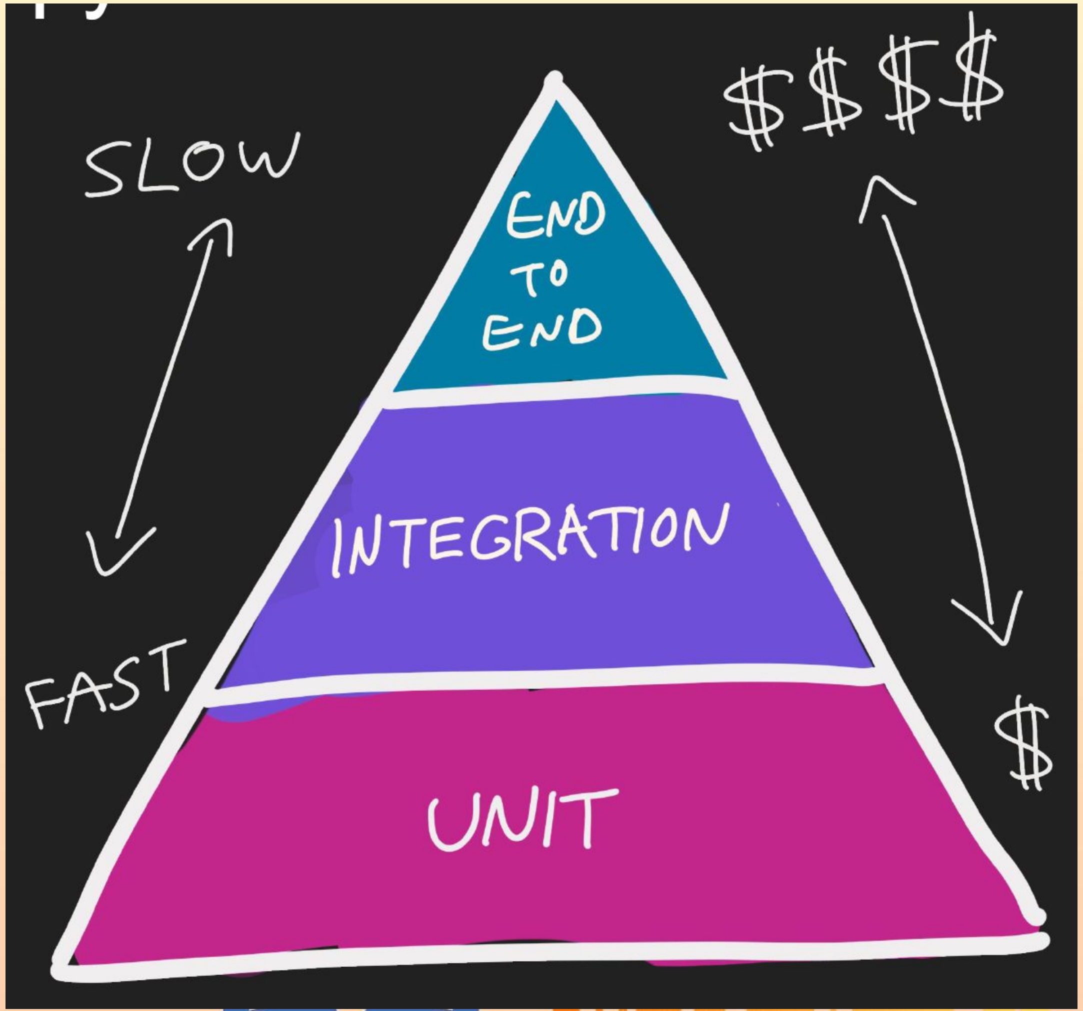
Bill Kennedy : SEA/-7
@goinggodotnet

Unit Test: Testing a single unit of code. In Go that's a package.

Integration Test: Testing two or more units of code. In Go that's a code path that runs through two or more packages.

Test Strategies

- E2E Test
    - uses `ginkgo` library
    - test a real cluster
    - black-box testing
    - can be promoted to conformance test
    - should only test *limited* happy/negative paths of a feature
    - optional for alpha feature



Test Strategies

- **Enjoy writing tests 😀**
  - hard-working and impartial reviewer
  - key factor of an OSS's success

Test Strategies

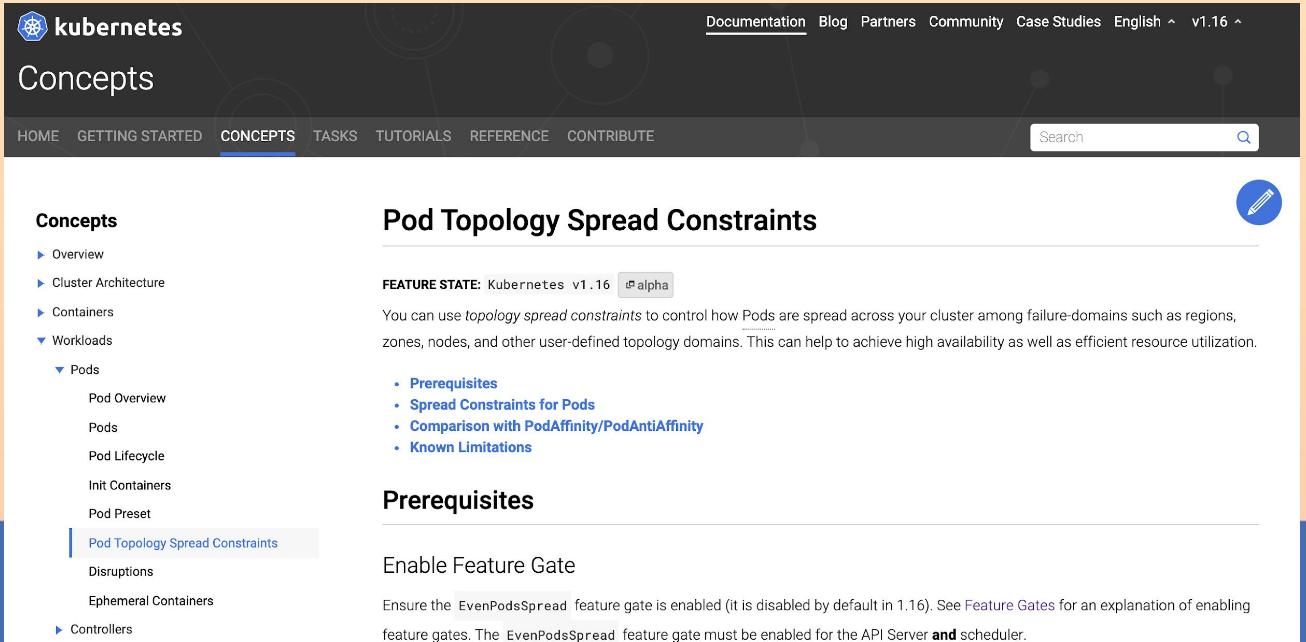
- Ad-hoc / Manual tests
  - forget it :)
  - how to test a feature manually
    - use kind
    - DIY hacking (only works for control plane components)
      - `START_MODE=nokubelet hack/local-up-cluster.sh`
      - `go build -o /usr/local/bin/hollow-node cmd/kubemark/hollow-node.go`
      - start N hollow-nodes in different terminals
        - `hollow-node --morph=kubelet --name=NodeA --kubeconfig=/var/run/kubernetes/admin.kubeconfig --kubelet-port=10261 --kubelet-read-only-port=10271`

Code Review

- Organize PRs properly
  - explain each section in the PR template, esp. the non-trivial part
  - take **RELEAE_NOTES** section seriously! (mistake of EvenPodsSpread)
- Be familiar with Golang [CodeReviewComments]
- For a huge PR, resolve comments in incremental commits, and squash them before merge
- Chase reviewers in a *reasonable* pace

Documents

- API inline documents
  - example => published at https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.16
- kubernetes/website documents
  - example => published at https://kubernetes.io



Pod Topology Spread Constraints

FEATURE STATE: `Kubernetes v1.16` alpha

You can use *topology spread constraints* to control how Pods are spread across your cluster among failure-domains such as regions, zones, nodes, and other user-defined topology domains. This can help to achieve high availability as well as efficient resource utilization.

- Prerequisites
- Spread Constraints for Pods
- Comparison with PodAffinity/PodAntiAffinity
- Known Limitations

Prerequisites

Enable Feature Gate

Ensure the `EvenPodsSpread` feature gate is enabled (it is disabled by default in 1.16). See Feature Gates for an explanation of enabling feature gates. The `EvenPodsSpread` feature gate must be enabled for the API Server **and** scheduler.

Tools/Commands

- **text searching tool:** `ag` (a.k.a the silver searcher)
- **git tools:** `tig, grv`
- **git commands and aliases** (some come up with oh-my-zsh)
  - switch to latest-used branch: `git checkout -`
  - squash/amend commits: `git rebase -i`
  - ban ~~`git merge`~~ as the “Merge...” commit pollutes the checkin history
  - searching/identifying: `git blame, git bisect`
  - aliases: `ggpush, glum, gbda, gco -b, gcp,` etc.
- **IDE**
  - Goland vs. VSCode

kubernetes CONTRIBUTOR SUMMIT SAN DIEGO 2019

Thanks!

Github: @Huang-Wei
Slack: @Huang-Wei
Twitter: @hweicdl