Kubernetes Working Group LTS

This Meeting

- Zoom: this meeting (passcode: 77777)
- This doc: **https://bit.ly/kubernetes-wg-lts-agenda**
- 45 minutes, bi-weekly at Tuesday at 7AM Pacific Time (convert to local timezone, calendar link)
- Based on https//bit.ly/k8s-sig-meeting-template

WG LTS Info

- **README:** https://git.k8s.io/community/wg-lts/README.md
- **Charter:** https://git.k8s.io/community/wg-lts/charter.md
- **Slack:** #wg-lts on kubernetes.slack.com
- **Mailing List:** wg-lts@kubernetes.io
- **Youtube Playlist:** https://bit.ly/k8s-wg-lts-videos(temp)
- **Project Board:** **https://github.com/orgs/kubernetes/projects/149/views/1**

WG LTS Chairs

Name	GitHub	Slack	Company
Micah Hausler	@micahhausler	micahhausler	AWS
Jordan Liggitt	@liggitt	liggitt	Google
Jeremy Rickard	@jeremyrickard	jerickar	Microsoft

How This Meeting Works

- This document is open for comment. We encourage suggestions for upcoming meeting topics. If there are none 24 hours before the meeting, we may propose canceling on our slack channel and mailing list.
- Action Items recorded in the meeting will be turned into Github issues and assigned to relevant project boards
- The meeting is recorded and posted to our Youtube Playlist before the end of the week. A host moderates the meeting, and we encourage hand-raising instead of interrupting.

Agenda

Template

Month Day, Year (recording)

Host:
Note Taker: TBD
Attendees:

-

Recurring Topics [timebox to N min]:

- (if you have any… here are some suggestions)
- Welcome any new members or attendees
- Project board review

Open Discussion [timebox to N min]:

-

August 26, 2025 (recording)

Host: Jordan Liggit
Note Taker: Bridget Kromhout
Attendees:

- Scott Dodson
- Rita Zhang
- Lubomir I. Ivanov
- Mo Khan
- Humble Devassy Chirammal

Recurring Topics [timebox to N min]:

- (if you have any… here are some suggestions)
- Welcome any new members or attendees
- Project board review

Open Discussion [timebox to N min]:

- [ritazh] Proposal: Community Security Patches for EOL Kubernetes Review updates based on feedback.
    - Feedback included: patches during embargo period would be more useful than later
    - No longer including embargo patches for out-of-support versions (but include out-of-support-for-12-months versions
        - Jordan: can we publish patches when not embargoed? (any CVE against K8s code where we're reporting and fixing) - Rita: yes

        - Jordan: what is the approach when a patch contributor doesn't backport to EOL versions - will we accept that from others?
        - Scott: whos in the owners file? liggitt: SRC in the owners file. SRC will tag CVE fixers as needed
- Jordan: do we require tests for the patch files? [notes in the doc about the details]
    - if can pick back cleanly, then yes. if not, don't include them
    - Rita: currently most CVE fixes include tests.
- Knowing how many CVEs did or did not have patches for the three recent EOL minors would tell us what kind of coverage we are getting - do we want to make it discoverable that there is a CVE we don't have a fix for?
- Mo: As an SRC member, we should start by just working with the CVE fixers to provide embargoed patches for the last 3 EOL versions. But not maintaining a new repo for all this.
    - liggitt: I dont think we should distinguish between embargoed or not
    - Rita: Without a new repo to persist the patch files, it will be hard for us to know if the fixer provided EOL patches can be applied cleanly since we do not persist the EOL branches anymore
    -
- Mo: what if we auto delegate the PR to sig leads to review/approve (automation needed to guide them, per Jordan)
    - Rita: do we need to get buy-ins from all the sig leads?
    - liggitt: No, it's best effort
- liggitt: SRC can work with CVE fixers to provide patches during the CVE process. Fixer can open the PR in this repo after it's public. If the patches are not available at the time of announcement, then SRC's job is done. After the CVE is public, anyone can push a PR to this repo and any sig lead has the power to review and approve or they can ignore.

July 29, 2025 (recording)

Host: Jordan Liggitt
Note Taker: TBD
Attendees:

- Bridget Kromhout, Jeremy Rickard, Rita Zhang (Microsoft)
- Scott Dodson
- Micah Hausler

Recurring Topics [timebox to N min]:

- (if you have any… here are some suggestions)
- Welcome any new members or attendees
- Project board review

Open Discussion [timebox to N min]:

- [ritazh] Proposal: Community Security Patches for EOL Kubernetes based on the suggestion: “Start with a set of best-effort patches in a folder per CVE” from the last meeting
-

[No meeting June 3 or June 17, 2025]

April 8, 2025 (recording)

Host: Jordan Liggitt
Note Taker: Zach Shepherd
Attendees:

- Jordan Liggitt
- Bridget Kromhout
- Humble Chirammal
- Micah Hausler
- Zach Shepherd
- Lubomir Ivanov
- Jeremy Rickard
- Peter Rifel
- Rita Zhang

Recurring Topics [timebox to N min]:

- (if you have any… here are some suggestions)
- Welcome any new members or attendees
- Project board review

Open Discussion [timebox to N min]:

- [micahhausler] Kubernetes LTS Subproject Proposal
    - Goals:
        - Have a common place for interested parties to contribute and maintain patches
        - Not “the regular Kubernetes” — no release artifacts, not the same support from SIGs
        - Intended to be separate group maintaining patches
        - Just k/k, not etcd, runc, containerd, etc. — that’d be up to independent distributors
        - Community location for patches to be maintained, rather than having multiple distributors doing this independently
        - Tags to agree on what is a point release, without all of the other release work that we do
        - Different maintainers (via different ownersfiles) to avoid asking every SIG to avoid maintaining things for a longer term
        - Analogous to what a set of distributors are doing today; they’re not coordinating with SIGs to get review, buy-in, etc., but in a shared place

    - Motivation is to have something in the Kubernetes project to maintain this
        - The alternative is to have this outside of the Kubernetes project, which is maybe possible, but seems less ideal
- [Jeremy] Would you see this happening within the existing k/k repo, or would this be a forked k/k?
    - Open to either, advantages to being k/k, but also disadvantages in allowing outside people to delineate, could be in a separate github org to keep repo name the same
- [Jordan] Scoping it to patch sets that users can contribute seems reasonable. Going beyond that gets blurrier and blurrier (and gets more expensive): testing it, merging it, release artifacts, docs, etc. Start with a set of best-effort patches in a folder per CVE?
    - [Micah] Trying to maintain the right level of ownership. Getting buy-in from every SIG just isn't viable. Differentiate the ownership of this from the core project owners.
    - [Micah] Merging, tagging, etc. would be owned by the LTS SIG. They might decide not to do tests.
- [Jordan] Wouldn't include dependencies, but what about vendor dependencies? (E.g., golang) Would want to avoid trying to ingest and merge that into a Kubernetes branch — especially if that's going to be happening everywhere downstream anyways. Does a "maintained branch" add value?
    - [Micah] Having a folder of patches is functionally equivalent, but there are benefits to merging patches
    - [Jordan] But the people who would be confused by a folder of patch sets are people we *want* to prevent from accidentally using a branch of merged changes and thinking they are completely supported — friction here would be a good thing.
- [Bridget] Some discussion of a separate group? Sub-project? Separate SIG? How would this be managed under Kubernetes governance.
    - [Micah] Unsure. Maybe it makes sense for it to be its own SIG, but the initial idea could be a sub-project. Unsure which SIG it'd be a subproject of.
    - SRC already coordinates changes to Kubernetes code to fix a security issue. Could be a directory owned by SRC. SIG Security owns security tooling. Could be a sub-project of either.
    - Do we have sub-projects off of committees?
    - [Micah] Ideally we want to keep it in the Kubernetes project.
- [Jordan] If we see this as a best-effort sort of thing, we need to be clear about that.
- [Micah] Expect close coordination between SRC and this group, align with CVE announcements
- [Zach] The "what we want to do" vs "how we want to do it" discussion - do we want to separate those out to ensure we're aligned on *what we want to accomplish* before we determine how?
    - [Jordan] The kernel of the proposal - share the work we're already doing on patches - is where we're aligned

    - [Rita] Reminder that whatever we decide to do, we do need a PR gate and therefore some CI/minimal tests
        - [Jordan] That may not be a given in that we may not be able to afford the cost and effort to keep green on more branches - what we merge might be a .patch file
        - [Rita] Even with patch files we need some measure of assurance of quality
        - [Micah] For example, “is it valid Go?”
        - [Jordan] CI we run rarely is likely to break when we do run it
        - [Rita] This aligns with SRC’s efforts on CI for private k/k builds
        - [Jordan] Collab on a patch has value; testing upstream still needs testing downstream (we could run “does the patch apply cleaning and does typecheck pass”)
        - [Jordan] For private k/k CI, we need to have comprehensive tests closely resembling k/k. But for an LTS/best effort, there can be a much reduced test coverage.
- [Zach] Whatever testing we land on, it will evolve over time - how do we empower whoever will own and maintain these branches to adjust things to meet their needs?
    - [Jordan] We need agreement on scope and deliverables of the project; mechanics can evolve
    - [Zach] How would we capture these constraints/non-goals?
    - [Micah] Okay with a fresh doc that focuses on the kernel
- [Mo] The way our LTS fork works, it’s a public repo, in a public org, with a public prow. Already have a set of folks who are doing the exact set of work: have E2Es break, fix those, etc. Could see those folks stopping that and then do this instead. We know the people exist; they do it now, it’s their job.
    - [Jordan] So have upstream maintained branches with tests, plus publicly maintained LTS branches, plus private testing?
    - [Mo] Yes, the AKS stuff needs to be private, but re-running the public test infra happens in public.
    - [Mo] If you tell existing SRC members that they have to maintain CI now, yeah… that’s not going to go well. But if you tell some people doing this work to do it in the context of the Kubernetes org, essentially grow the team.
    - [Jordan] Starting with the smallest viable approach seems safest. We overpromise. Things change, and then go red. Start with the smallest kernel and then build up and out.
- [Rita] Empathetic to all of this. This is a good forcing function to get more people helping out in key areas. We’re all doing this work, but not within the project. If we maintain things in a central way, we can get more people involved. Can start small and see what people show up?
    - [Jordan] The more resources that get devoted to it, the more pressure there will be to declare it as supported.
    - [Rita] Don’t have to promise the world. If companies want to depend on it, that’s there prerogative.

        - [Mo] Would it make people feel better to say any infra costs won't be borne by the project? Could repurpose existing AKS prow instance.
        - [Jordan] Experience with out-of-project CI hasn't been great. But if this is all best-effort anyways, maybe the consequences are small.
        - [Zach] Want to avoid failure mode where the new in-project LTS repo is not sufficient to replace the vendor-specific public LTS forks. If the in-project LTS repo is in addition to vendor-specific public LTS forks, then it seems like we can't count on help from the folks who maintain the vendor-specific public LTS forks.
- [Rita] Clarifying that this would be a fork, not new branches.
- [Jordan] Folder of best-effort patches seems useful, and can be successful. The project hasn't demonstrated that it can be successful at something more than that.
- [Micah] Will try to clarify based on discussion what the goals and non-goals are. May propose a new SIG as it's a code-owning body with different owners.
    - [Jordan] Hard to imagine a charter that doesn't usurp some existing group.
    - [Jordan] Would be happiest with putting it close to SRC. Already handling coordinating responses for vulnerabilities in Kubernetes code. Extra bit is just adapting this to some older branches — which seems like a natural extension.
    - [Rita] And downstream folks will take the patches, apply downstream, and report back on downstream testing results.

March 25, 2025 (recording)

Host:
Note Taker: Zach Shepherd
Attendees:

- Kat Cosgrove
- Zach Shepherd
- Humble Chirammal
- Benjamin Schimke
- Scott Dodson
- Rita Zhang

Recurring Topics [timebox to N min]:

- (if you have any… here are some suggestions)
- Welcome any new members or attendees
- Project board review

Open Discussion [timebox to N min]:

- [micahhausler] draft proposal for discussion on an LTS subproject
    - https://docs.google.com/document/d/1MzSNg2hFEFRnkRyCbEKpt6NFJzBD7FXpfRHY8bNjA_g/edit?tab=t.0
    - Canonical would want to support and contribute to such efforts proportional to their size.

- [Antonio] Kubernetes LTS is just part of the problem. What about golang, etcd, containerd, etc.? We're concerned about security, so we need to consider the dependencies.
- [Scott] Would we have a common golang fork as well?
- [Jordan] Collaborating on coming up on non-conflicting backports is the lightest weight thing. That helps share resources a little bit. At the other end of the spectrum is something with forked golang and everything gets bumped and we have test-infra for everything. But that doesn't seem maintainable. Where on the spectrum is this proposing?
- [Rita] MVP would be the bare minimum: backport patches for Kubernetes CVEs. Every vendor is doing the backporting for other dependencies as well though. We're all doing it anywhere.
- [Jordan] Different timeframes; not all 12 years.
- [Jordan] Even if we did testing centrally, we'd all need to do downstream testing as well. Does the upstream testing provide value?
- [Scott] As the go backwards compatibility stuff becomes more widely used, we see libraries that don't support anything but the latest golang. Would we continue golang updates in this fork? What RedHat is seeing is that the backwards compatibility is sufficiently robust to update to latest version of go.
- [Antonio] We have cases where golang breaks, right? (Breaks in terms of behavior.)
- [Jordan] Treated as regression, they're good about fixing it. (Promise of runtime compatibility for a minimum of 2 years, which covers a "2 year LTS window" — but not a 5y or 12y window. Have seen them responsive to keeping along generally.)
- [Antonio] What about longer than that?
- [Jordan] Can't let LTS window go longer than that of core dependencies that we don't have a way to maintain.
- [Jordan] Consolidate ownership and change approvals seems counter to the things that have made the project successful. We have rabbit holes of complicated components. Some of those areas are where we see vulnerabilities. Skeptical of consolidating ownership of complicated fixes under SIG Arch; in reality, we would have to be pulling in TLs and reviewers/approvers from every random SIG. (And would SIG Release be more natural? Involved in go bumps. But they also lack the review expertise: they coordinate.) Feels like central agreement to distributed work.
- [Rita] Perhaps the subproject would reduce the burden on other SIGs, but not eliminate it.
- [Kat] SIG Docs item is also a Release Docs item.
- [Jordan] There are trivial backports. But then maybe this doesn't provide value. And there are complicated backports. But then you want the local expertise to deconflict, etc.
- [Antonio] Do we have anyone from Canonical?
    - [Ben] Yes. One of the engineers.

    - [Jordan] May want to watch last time's recording.
- [Zach] How do the existing distributions handle this without involving all the SIGs today?
- [Scott] Everyone does it unilaterally. Getting the 6 groups to agree may be the challenge.
- [Rita] This could be a way for Canonical to provide value. Lots of backporting experience.
- [Jordan] One of the goals has been to make the normal go bumps pretty trivial, non-complicated. So you can just bump the go version you're building with. But some bump go versions. Some maintain patched go versions. Pros/Cons. Two approaches don't go together; need to pick a pattern.
- [Jordan] For complicated backports: try to minimize time period (10-12 months for GKE) and then as backports arise, ideally when there is a security backport the code is still hopefully pretty similar. When a rewrite is required, that's where sharing would provide the most value. If we wanted to start with that: when there's a Kubernetes CVE, get volunteers to help re-write those for a few more branches. Best effort. Avoids commitment to the toil: bumping dependencies, maintaining branches, etc.
- [Jeremy] Just sent to distributors list?
- [Jordan] Publish on the same timeline, using the same mechanism, as other security fixes. Don't merge. Don't run CI.
- [Jeremy] Cherry-pick PR that doesn't merge?
- [Jordan] Yeah
- [Zach] If this were the focus, where should the sub-project live?
- [Jeremy] SRC, for coordination
- [Rita] Just sharing the patch is a good first step. But what's the right amount of vetting/approval to make sure it's useful?
- [Jordan] Wasn't clear from discussion last time that all of the hard problems were fully solved by the Canonical offering. And from personal experience, skeptical of backports of complicated Kubernetes security fixes being backportable. Concerned about maintaining community branches for that long with community staffing processes.
- [Mo] If RedHat can do for RHEL, and Canonical can do this for Ubuntu, why can't someone do this for Kubernetes?
- [Antonio] It's those companies' business.
- [Mo] Do we actually need SIG Leads to make sure this is done correctly? For Linux, expertise has diffused.
- [Jordan] But within the Kubernetes project, the SIG Leads are where expertise lives.
- [Mo] Maybe CNCF is the right scope for this proposal? Completely detached. Just CNCF as a governing body to provide a way to break ties. When it's time to backport something crazy, it's just on that project to figure it out; not on Kubernetes SIGs to help. In the same way that when K3s has issues, it's not on K8s SIGs to help.

- [Antonio] Aren't you inevitably creating a dependency on people who aren't looking for more hard dependencies?
- [Jordan] Something else under CNCF owning this seems really weird. Separate group making different decisions about support, dependencies, etc. The further you get from the people who actually understand, the less likely this is to be successful.
- [Rita] Trying to balance between reducing burden for TLs, but at the same time we want them to be able to provide input when it's needed.
- [Jordan] Feels like we're saying we need the expertise, but can't count on it?
- [Rita] Can we have a group handle the easy stuff without burden people who are really stretched?
- [Jordan] Even go bumps take a lot of effort. If there's lots of effort or energy or interest in expanding that work, would like to see more staffing in the area of the project that does that work. Shore up the areas of the project that are understaffed.
- [Rita] Could be more incentive to attract funding if this is the means to get extended support. But without committing to that as a community… that's where the funding comes from. Saying that the staffing goes towards extended support is where the funding comes from.
- [Jordan] The more formal the commitment is, the more real the plan needs to be.
- [Rita] Need to set boundaries.
- [Jordan] The project is at that point now. The vendors are taking on significant burden.
- [Zach] Seems like there's a spectrum. At one end, you have the SIGs themselves owning extended support. At the other end, you have vendors doing their own thing on their own. In between, you have options like a subproject under SIG Architecture, SIG Release, or SRC. Or a project in CNCF.
- [Kat] This proposal is getting very close to the Kubernetes project just supporting versions for longer. Is that what's happening here? There's no maintenance of the docs branch for older versions. Docs may become useless or actively misleading. Could stop removing old branches from the website, maybe with a warning, but that feels bad.
- [Rita] Expected docs to be out-of-scope.
- [Kat] Not having the docs doesn't feel like the right path. But SIG Docs doesn't have the resources to maintain it.
- [Antonio] Pretty sure that each vendor's fork has some code that's incompatible with the others.
- [Jordan] Look forward to having this discussion again when Micah can join us. Feels way too close to the project supports more minors.
- [Jordan] Coming back to something to start with: when there is a particular patch for a particular issue, share. Smaller of an initial step. Might save some time. Clearer that this is just a best effort, not project support for more minors. Establish a process for sharing. Under SRC. Potentially get reviews from the fix

team, who often have to do that work anyway. Not a promise for every CVE to come with this, not a promise for testing.
    - [Zach] Proposal?
    - [Jordan] Not yet. Could pare the proposal down to match this or write an alternative. Could go into the SRC playbook as instructions on where and how to share the patchsets, if/when people show up to collaborate on it for an issue.

Feb 25, 2025 (recording)

Host: Jordan Liggitt
Note Taker: Zach Shepherd
Attendees:

- Konstantinos Tsakalozos - Canonical
- Vyom Yadav - Canonical
- Benjamin (Ben) Schimke - Canonical
- Marcin (Perk) Stozek - Canonical
- Scott Dodson - Red Hat
- Zach Shepherd - VMware by Broadcom
- Rita Zhang - Microsoft
- Mo Khan - Microsoft
- Bridget Kromhout - Microsoft
- Paco Xu - DaoCloud
- Micah Hausler - AWS

Recurring Topics [timebox to N min]:

-

Open Discussion [timebox to N min]:

- Annual report due by end of month. Jordan will be working on this in the next few days, but if anyone else has time to add links, etc. that would be appreciated.
    - Draft open at https://github.com/kubernetes/community/pull/8300 (editable version at wg-lts 2024 annual report draft)
- [kjackal, ritazh] https://canonical.com/blog/12-year-lts-for-kubernetes : what and how Canonical is backporting to its LTS k8s components and if there's an opportunity to collaborate
    - Scope
        - Starting with kubernetes 1.32
        - CNI, ingress, gateway API (Cilium)
        - Load balancer (Metal LB)
        - Local storage (Local PV from OpenEBS Local PV)
        - Metrics Server
        - DNS (CoreDNS)
    - Everything packaged inside a snap, based on an LTS Ubuntu

- Canonical does not specify exactly where these core features are provided from; Canonical is preserving the option to swap one implementation of a capability for another.
    - Question(liggitt): do you hide or prevent use of the surface area of the implementations (APIs, etc) to preserve ability to swap them out?
        - No, don't hide anything. This is intentional: want to stay as close to the upstream dependencies. But want to preserve the ability to switch in the future.
        - In theory, the ability to switch out an implementation sounds nice. But in practice, it seems impossible to actually switch things out? Users could be reading from specific APIs, etc.
        - LTS doesn't cover backporting features, API changes, etc.
        - Expect that users understand that API changes may be necessary when upgrading between LTS versions.
        - Intend to keep the upgrade path as smooth as possible.
- Support is 12 years, under the Ubuntu Pro umbrella. Free for first 5 nodes for everyone.
    - Won't be supporting every Kubernetes version for 12 years. With Ubuntu, you get an LTS every 2 years, so the same cadence will apply: one Kubernetes version every 2 years (which will then have a 12-year support period).
    - Question: How will users upgrade from one LTS version to the next? (N→N+6?).
        - A: we have to go through intermediate releases during LTS-to-LTS upgrade. If at a very late stage an LTS-to-LTS upgrade will go through intermediate versions that are not supported for 12 years, so could be unpatched after a shorter support version.
            - The intermediate versions won't be supported for 12 years, but will be supported long enough to provide some overlap between LTS versions (say a year) so that the path the user needs to follow is through fully patched releases.
            - However, not all releases will be LTS. So if it's been like 10 years on one LTS version, the intermediate versions won't be supported by Canonical at that point/won't be patched.
            - May be essentially two modes of consumption:
                - Hop from LTS to LTS as each comes out.
                - Stay on one version a very long time, then plan some bespoke path to get updated. If you're 12 years late, you may prefer to just redeploy at the N+36 version.
        - (liggitt) Do you see issues with skip version upgrades of packaged features like Cilium / LocalPV? (e.g. Cilium documents "The only tested rollback and upgrade path is between consecutive minor releases")

        - A: plan is to not skip versions
    - (sdodson) Is this security only or how are we handling a situation where a critical functional defect (e.g. dataloss bug) is fixed in 1.32 but you have to upgrade through unpatched 1.33-1.35 where that regresses?
        - No, but ensure that the upgrade will continue.
    - Q: is there orchestration of the multi-step upgrade
        - A: not at this time, there are tools that can do orchestration
- Question(liggitt): What Go version is being used to build these? Updated go versions or Canonical-maintained (security-patched) older go versions?
    - go is just another package maintained by canonical, expectation is to generally keep that go package patched
    - Multiple options:
        - The preferred option is to try to patch the version already used. Try to use the same go version. Switching may have side effects hard to catch in tests.
        - Rare cases may require that a package (or go — it's just a package) update be backported from one LTS to the previous.
    - (liggitt) Go has bi-annual releases, and has a higher volume of security fixes than in Kubernetes (maybe 4-5x).
    - Canonical already has a go snap, and may have to do some of the backporting. But that'd be another team. (The go runtime is owned by a "foundations" team)
        - Today, the golang snap isn't patched but the debian package is. Prioritize critical and high vulnerabilities; medium is best-effort. Intend to reflect this into the snap.
- Question: how do you plan to guarantee compatibility for components like Cilium relying heavily on the Linux Kernel ?
    - Ubuntu has 5 years of public support, but under Ubuntu Pro you get 5+5+2. So expectation is that you're not switching kernel within the 12 year period of the Kubernetes LTS.
- Question: any plans to backport any other CNCF projects?
    - Not aware of any, but there may be other teams with other plans.
    - Istio? this is mentioned in the blog: "Canonical will also provide standardized container images of popular Kubernetes ecosystem services such as Istio"
        - There's some set of components: Istio, cert-manager, Harbor, etc. in scope for Ubuntu Pro.
    - Except for these items, packaged with the LTS release:
        - CNI, ingress, gateway API (Cilium)
        - Load balancer (Metal LB)
        - Local storage (Local PV from OpenEBS Local PV)

            - Metrics Server
            - DNS (CoreDNS)
        - Ability to backport any CNCF project for customers under Everything LTS (Not by default, on-demand paid service)
- Canonical has experience delivering LTS. Practically, had to onboard Kubernetes packages into existing Canonical support mechanisms.
    - If there's a scan or customer that identifies something that must be fixed (maybe a bug or a vulnerability), anywhere in the dependency tree:
        - The affected repository is forked.
        - Apply a patch (could be easy: backport a patch, could be something harder: implemented by engineers).
        - Go through testing of affected components. This depends on what the components the bug affects.
        - For Kubernetes, will re-use relevant parts of the upstream test suite. Probably won't redo scale testing, for example. Details depend on the specifics of the issue.
        - Produce a new release, in the form of a snap. (Fat package that includes all dependencies; reduces number of targets that need to test)
    - And then upgrades happen…
- Very interested in an LTS-to-LTS upgrade. With 2 year cadence and upstream version skew: can't jump through releases. When upgrading from LTS-to-LTS, traverse through the intermediate releases. Intermediate releases will be supported and patched for a reasonable amount of time to handle the LTS-to-LTS path. If a user waits until very late to go through the LTS-to-LTS path, they may go through intermediate patches that have some regressions.
- (liggitt): Is the user navigating the LTS-to-LTS operation? Is LTS-to-LTS single user-facing operation? Sometimes it's very hard to make something that works with two Kubernetes versions two years apart.
    - The snap does not provide orchestration.
    - Canonical does offer orchestration with Cluster API and Juju.
    - Some of this is going to be user-specific, relating to how they've deployed their nodes.
    - Expecting that during the upgrade, at intermediate points, the user may need to do some incremental steps to manage/update their workload. Potentially at multiple points.
    - Unless Kubernetes comes up with a better strategy for skip-version upgrades, which Canonical would want to adopt.
-
- [micahhausler] For future discussion - draft proposal for discussion on an LTS subproject
    - https://docs.google.com/document/d/1MzSNg2hFEFRnkRyCbEKpt6NFJzBD7FXpfRHY8bNjA_g/edit?tab=t.0
    - Canonical would want to support and contribute to such efforts proportional to their size.

-

Feb 11, 2025 (recording)

Host: Jeremy R
Note Taker: Bridget K
Attendees:

- Scott Dodson
- Siyuan Zhang
- Wenjia
- Jeffrey
- Humble Chirammal (VMware by Broadcom)

Recurring Topics [timebox to N min]:

- (if you have any… here are some suggestions)
- Welcome any new members or attendees
- Project board review

Open Discussion [timebox to N min]:

- [liggitt - 5 minutes] FYI: wg-lts annual report due - https://github.com/kubernetes/community/issues/8281
    - Draft open at https://github.com/kubernetes/community/pull/8300 (editable version at wg-lts 2024 annual report draft)
    - Main gap is giving updates to sponsoring sigs (via meeting or mailing list update, etc)
        - SIG Architecture
            - Compatibility version KEP proposal (intersection of architecture and LTS)
        - SIG Cluster Lifecycle
        - SIG K8s Infra
        - SIG Release
        - SIG Security
        - SIG Testing
    - What to tell sponsoring SIGs: our recent focus on the compatibility work
        - Jordan volunteers to start the doc - draft the message to SIGs, summarizing current focus to do not exactly LTS releases but rather compat versions/upgrade experience improvements.
- [liggitt - 10 minutes] compatibility version / emulation version work update
    - 1.33: dev process / testability improvements - https://github.com/kubernetes/kubernetes/pull/129416
    - 1.33: tweaks to behavior for API progression - https://github.com/kubernetes/enhancements/pull/5038

    - sweep of features / impact to expand to **node / network components** still pending (exploratory)
    - [siyuanfoundation] presented compatibility version in sig-node meeting, they are generally ok with it, as long as the feature cleanup at N+3 releases after GA is part of PRR/KEP review/milestones to track. Recent node GAed feature clean ups: [Public] Compat Version for Node
    - [liggitt] control plane features, dev exp as focus (for 1.33)
    - [liggitt] Our API versions get decorated with version being served - after 3, deprecated warnings, and after 6 releases, auto-removed - maybe we can look into something similar.
    - [bridget] auto-removal may lead to unpleasant surprises
    - [liggitt] Announcements around GA graduation is when people are paying attention - perhaps release-blocking jobs but not auto-removal.
    - [liggitt] a summary of this direction/these thoughts would be valuable to bring SIGs.
- Feb 24 we'll decide if we will have a Feb 25 meeting (Jeremy R & Bridget will miss Feb 25; Jordan may run it.)
- [liggitt] getting Go updates into older release branches can be tricky
    -

Jan 28, 2025

- Only one attendee other than host, agenda sent to mailing list and deferred

December 03, 2024 (recording)

Host: Jeremy Rickard
Note Taker: Bridget Kromhout
Attendees:

- Zachary Shepherd - engineer at VMware by Broadcom
- Jordan Liggitt
- Bridget
- Jeremy

Recurring Topics [timebox to N min]:

- (if you have any… here are some suggestions)
- Welcome any new members or attendees
- Project board review

Open Discussion [timebox to N min]:

- KubeCon follow up
    - Mo's proposal: [PUBLIC] wg-lts: proposal to increase release lifetime with r…

- Followup to discussion Jordan and Jeremy had with Ben the Elder at contrib summit - Is focusing on skip upgrade + compatibility mode a better long term support model?
- Contributor summit notes drive: Session Notes
- LTS notes: wg-lts: cve support period
- Jeremy: Reactions from contrib summit: concerns about backports, CI lacking on private fork - considering this to be an additional burden for SIGs at the point where maintaining tests is already a challenge
- Jordan: And backporting gets harder the farther back you go
- Zachary: have we gamed out trying to backport recent impactful CVES farther back, to assess cost/risk?
- Jordan: LTS motivations are often around compliance, not just the big risks.Concretely, half of fixes we would be trying to backport are trivial, 25% require some tweaks, and 25% would require quite a rewrite.
- Zachary: and how much SME versus golang expertise is useful?
- Jordan: lots of subject matter and it decays over time - also people want to linger on old versions due to potential disruption
- Jeremy: Maybe “actively supported releases” versus “maintenance releases”
- Jordan: while blue-green and perhaps in the future skip-version would help, people currently do one upgrade near the far end.
- Zachary: qualifying their software on a given version could take months, followed by the actual upgrade process (also applies to ecosystem software - non-K8s)
- Jordan: API stability work should now mean that qualifying an upgrade should be much simpler/faster than it was
- Zachary: the ecosystem is a factor here too: many projects document some sort of version compatibility between their versions and Kubernetes versions, and ship new minor versions that “add support” for the new Kubernetes version
- Jordan: perhaps saying “we’re using these APIs at this version or higher” instead of “K8s version”? (Bridget thought that might be difficult to digest, but we would still say which K8s version, but would make it easier for people to know that a later version also supports it)
- Jordan: emulation version would also make it easier to make in-place upgrades possible. Need to follow up with sig node and sig network. If we could add skip-level emulation it could bridge this further.
- Jeremy: also making cluster lifecycle easier to support this
- Jordan: because this work is already in progress, it has more benefits including a path to skip-upgrades.
- Zachary: making emulation work may incur similar amounts of work as backports
- Jordan: this will encourage upgrade tests - which is a beneficial use of upgrade costs
- Jordan: this approach is something that’s used widely across software projects in general, and was what golang in particular decided to do when asked for LTS
- Jordan: after the release gets out may be a good time to assess impact of the compatibility version approach

- Bridget: topic for December 17th meeting?
- Jordan: use December to do the research, then queue up conversations for January; expecting light attendance in December meetings, cancelling December 31st
    - AI: research last 1-2 years of feature graduations / cleanup in node / network (similar to https://github.com/kubernetes/enhancements/tree/master/keps/sig-architecture/3935-oldest-node-newest-control-plane#evaluate-previous-control-plane-releases)
    - AI: start threads on node and network mailing lists once initial sketch of research is done or in progress (~Dec 17th)
    - AI: get onto node / network SIG meetings for mid-January (mention in list threads in Dec and again in Jan to get attendance for discussion)
- Bridget: January 14 vs 28?
- Jordan: the ask is to not delete code for gates graduating in v1.33, so sooner is better
- Jordan: there may be a central place where we can catch feature gate deletions
    - AI: make sure central gate graduation / tracking tests flag deleting gates earlier than 3 versions post-lock

Oct 22, 2024 (recording)

Host: Jeremy Rickard
Note Taker: Bridget Kromhout
Attendees:
- Scott Dodson
- Jordan Liggitt
- Dhanush A
- Lubomir I
- Paco X

Recurring Topics [timebox to N min]:
- (if you have any… here are some suggestions)
- Welcome any new members or attendees
- Project board review

Open Discussion [timebox to N min]:
- [Jeremy] Contributor Summit Session(s)
    - wg-lts: proposal to increase release lifetime with regards to CVEs (Jeremy & Mo)
    - Kubernetes, Upgraded (Han & Joe)
        - Need to get these scheduled at different times from each other - Jeremy is reaching out to Summit staff
- [Jeremy] Discuss communicating and approving Clarify cherry-pick candidates to fix Strengthen Cherry Pick Guidance · Issue #7634 · kubernetes/community · GitHub
    - Let's look for any other place where we need to tell people to check

    - https://github.com/kubernetes/community/blob/master/contributors/devel/sig-release/cherry-picks.md is canonical location

- [Jordan] Updated Kubernetes patch release regression/bugfix rate and captured patch regression stats
    - While 1.28 and 1.29 were pretty smooth, 1.30 had a number of regressions (but some were indirect/client-api-doc-related)
    - Kubeadm refactors broke - but were fail-fast and simple to resolve
    - But also some issues in patch releases (perf, panics)
    - We're trying to tighten the backport policy in Clarify cherry-pick candidates because of these sorts of regressions.
        - Jeremy will email the dev list
- [Jeremy] Potentially cancelling Nov 5 meeting due to lead-up to KubeCon - will update in Slack

Sept 24, 2024

(Team Microsoft at an offsite)
Empty agenda, no meeting

Sept 10, 2024 (no recording, only a brief call that didn't turn into a meeting)

Host: liggitt
Note Taker: Bridget, Vince
Attendees:

- Jordan Liggitt
- Bridget Kromhout
- Mo Khan
- Vince Power
- Lubomir Ivanov

Action items for [PUBLIC] wg-lts: proposal to increase release lifetime with regards to C…

- Discuss with other SIGs (volunteers to take this to key stakeholder SIG meetings, SIG leads meeting, etc)?
- Answer open questions
- Pin down costs (to project, to ecosystem, CI dollars, people hours, etc)

August 27, 2024 (recording)

Host: jerickar
Note Taker: Bridget
Attendees:

- Jeremy Rickard
- Jordan Liggitt
- Bridget Kromhout
- Mo Khan
- Micah Hausler
- Marko Mudrinić
- Scott Dodson
- Tim Pepper

Recurring Topics [timebox to N min]:

- (if you have any… here are some suggestions)
- Welcome any new members or attendees
- Project board review

Open Discussion [timebox to N min]:

- [Mo] [PUBLIC] wg-lts: proposal to increase release lifetime with regards to CVEs
    - (continued)
    - [Mo] Planning on nuanced decision about what makes a backport reasonable (CVSS and difficulty/risk)
    - [Mo] Hoping we adopt this in a manner that is purposefully non-retroactive (only applying to the current and future versions, not old versions).
    - [Jordan] we should ensure we know which SIGs are most impacted by support period extensions (for example, SIG node)
        - [Jeremy] I believe we (Azure) have some analysis of this from when we examined LTS internally
    - [Micah] What about vulns in removed features?
        - [Jordan] That does need to be in scope.
        - [Tim] This is needs to be clarified
    - [Tim] What happens if a CVE fix is “remove feature”? This is more change to an old branch than may be expected.
        - [Jordan] We backport the way to turn off the feature without changing defaults.
        - Scope of bug bounty https://hackerone.com/kubernetes (although budget is unknown currently)
    - [Marko] Changes in tests - less-frequent tests would lower costs - once forked for a minor, should not need to change
        - [Bridget] I’m concerned tests will decay over time.
        - [Jordan] Occasionally they’ve changed for example with go version changes (look at 14-month history of job configs, and further back over time)
        - [Jeremy] Infra moves have also affected things

        - [Jordan] Let's ensure specific named groups verify test updates.
    - [Jeremy] Release costs include some serial image promotions (run into rate limits) which require stretching across time zones - so hosting isn't as much as maintainer toil.
        - [Marko] agree about needing to improve image promotion process
        - [Marko] Is it really useful if we don't publish releases?
        - [Jeremy] The effort lies in getting the branches up to date; cutting the binaries isn't the costly part.

July 30, 2024 (recording)

Host: liggitt
Note Taker: Bridget
Attendees:

- Jordan Liggitt
- Bradley Ascar
- Bridget Kromhout
- Mo Khan
- Micah Hausler
- Marko Mudrinić
- Benjamin Elder

Recurring Topics [timebox to N min]:

- (if you have any… here are some suggestions)
- Welcome any new members or attendees
- Project board review

Open Discussion [timebox to N min]:

- [mo] [PUBLIC] wg-lts: proposal to increase release lifetime with regards to CVEs
    - Started this doc some time ago - have now responded to a number of the comments
        - Jordan: supporting a minor longer requires input from every sig.
        - Mo: So we'd need a leader from each sig to sign off on the kep - create a link in every sig's folder so every sig lead can be tagged.
        - Marko: if downstream folks aren't upstreaming patches it will be unsustainable
        - Ben: the owning sigs will still need to vet changes, same as always
        - Ben: tests breaking or flaking will still matter in old branches that we're supporting long term - may impact skew support
        - Jordan: if the real goal is "I want to upgrade my cluster once a year and I need a year to do the upgrade" then we also need skip-version
        - Discussion about the tension between features-being-added and allowing-backwards-compat
        - Additional comment updates as noted in the doc

- [liggitt] go 1.23 compatibility mode improvements landing
    - The updates in 1.21 didn't work as desired, but they improved the directive to ensure that you won't accidentally get bumped up. We'll miss it for the next release but will be able to pick it up in January.
    - Go version updates after 1.23 will have this new operational mode
- Add your suggested topic here and move this to the line below [firstname lastname, email or @slackname]

June 18th, 2024 (recording)

Host: Jeremy Rickard
Note Taker: Bridget Kromhout
Attendees:

- Scott Dodson - Red Hat
- Brad Ascar - Redis
- Marko Mudrinić - Kubermatic
- Add your names here

Recurring Topics [timebox to N min]:

- Welcome any new members or attendees

Open Discussion [timebox to N min]:

- Annual Report Feedback Add initial draft of wg lts annual report by jeremyrickard · Pull Request #7910 · kubernetes/community (github.com)
    - SIG Updates: Volunteers? (add your name if you'd like to volunteer to visit a SIG meeting)
        - Architecture (Jordan)
        - Cluster Lifecycle
        - K8s Infra (Jeremy)
        - Release (Jeremy)
        - Testing
        - Security Response (committee) - no meeting so could be email/notification to another meeting
    - WIP doc to gather update material - (update our sponsoring SIGs list)
        - WG LTS 2024 Update for Sponsoring SIGs
- [PUBLIC] wg-lts: proposal to increase release lifetime with regards to CVEs
    - Any feedback?
        - [Jeremy] We need to consider monetary as well as time/staffing costs
        - [Jordan] Historically about 50% of go version updates have some security content
        - [Jordan] If we don't provide upstream builds, it will be confusing (Jeremy agrees)
        - [Scott] agree - we need to agree on a version that is a specific patch - that's part of the value

        - [Jordan] let's iterate on this doc and answer some of the more obvious questions with the tradeoffs
        - [Brad] would be useful to see the reason why if we skip a CVE (like if it's not executed)
        - [Jeremy] Perhaps a vex document for vetting vulns
        - [Bridget] We definitely need to weigh backports vs risk
        - [Jordan] we have seen regressions and we need to watch out for this (for example, with xnet)
        - [Jordan] this proposal didn't talk about ripple effects for projects that support specific numbers of old K8s versions to cover (eg Gateway API) - Jeremy added to the open questions.
        - [Jordan] Control-plane/node skew will be affected by the expansion of support (not necessarily will overlap anymore)
        - [Jeremy] To do before broad dissemination of this doc: cost estimates
    - Should we roll this into SIG Updates?
        - [Bridget] I think we should list it, but it doesn't fit under the 2023 scope

May 21st, 2024 (recording)

Host: Jeremy Rickard (Microsoft)
Note Taker: Bridget Kromhout (Microsoft)
Attendees:

- Mo Khan (Microsoft)
- Nick Young (Isovalent)
- Vince Power (Dell)
- Krzysztof Wilczyński (Red Hat)
- Arka Saha (VMware by Broadcom)

Recurring Topics [timebox to N min]:

- (if you have any… here are some suggestions)
- Welcome any new members or attendees
- Project board review

Open Discussion [timebox to N min]:

- [mo] security releases proposal
  [PUBLIC] wg-lts: proposal to increase release lifetime with regards to CVEs
    - This proposal was originally discussed in the SRC meeting (Rita and Micah were present) - no objections in that forum.
    - Mo's assertion is that the security-backport work is already happening, and consolidating the effort in the upstream releases would be a savings for all concerned.
    - "Does it have a CVE number" offers a clear rubric that is not subjective.
    - Two-year cycle acknowledges the reality of patch cycles
    - Binary releases would serve the broadest community of users
    - Cloud provider credits will be relevant for the testing needed

        - [Jeremy] Necessary action item: work with k8s-infra on cost projections - we have done a bit of these “additional releases” for go version bumps. Likely need extra credits to add to the spend supported currently.
        - [Vince] Is this proposal only for months with CVEs, or monthly? Let’s figure out cadence.
            - [Jeremy] We may only want to create a new release with changes, but the build infra doesn’t cost much more than just ongoing tests.
            - [Nick] how will this affect the sig-release workload in a month with additional releases?
            - [Jeremy] we may need to split the release work over additional days
            - [Mo] Release team can trigger the builds of old versions, keeping costs down if builds aren’t needed.
        - [Jeremy] will embargos affect our ability to release in a timely fashion?
            - [Mo] we try to ensure that the order is embargo list -> PRs -> release
        - [Nick] When this was discussed in the past, this came down to cost - CI costs but also manual work, so we want to improve the tooling for the release automation.
        - [Nick] “No CVE? No backport!” is the cleanest way to start
        - [Mo] It’s key that this won’t apply to anything *already* out of support.
        - Next steps: Jeremy and Mo talking to stakeholder SIGs (potentially to be followed by a KEP)
- Action item follow ups
- Add your suggested topic here and move this to the line below [firstname lastname, email or @slackname]

May 7th, 2024 (recording)

Host: liggitt
Note Taker: bridgetkromhout
Attendees:

- Jordan Liggitt
- Scott Dodson
- Bridget Kromhout
- Mo Khan
- Marcin Franczyk
- Arka Saha

Recurring Topics [timebox to N min]:

- (if you have any… here are some suggestions)
- Welcome any new members or attendees
- Project board review

Open Discussion [timebox to N min]:

- KubeCon EU 2024 - Contributor Summit - LTS Discussion Notes
    - LTS discussion recorded but also summarized by Mo in the doc

- [Mo] What is LTS? - if it's more than security fixes, people start disagreeing about whether a change is worth it. If "there is a CVE" is the bar, then that's smaller.
- [Mo] What happens when we need to jump to the next version? (Right now, the vendors have a lot of upgrade tests - seeing that in upstream would be helpful). Antonio brought up the lack of upgrade tests as well as the lack of feature interaction tests.
- [Jordan] e2e testing tries to cover turning on all the alpha features
- [Bridget] testing all possible combinations is untenable
- [Mo] "LTS brings value without cost" but Jordan points out that the ecosystem support in, for example, gateway API won't be there (the costs are there but aren't as obvious)
- [Jordan] if you only depend on core components, then extra security patches for ten months is fine, but ecosystem things won't necessarily keep up support-wise.
- [Scott] Can ecosystem projects support skip-level more than core kube does? [Mo] In a number of cases, yes.
- [Jordan] Minor control-plane boundaries are still a problem - things around the cluster tie in their version bumps - "A lot of other stuff changed too" at a minor version change.
- [Mo] Who pays is an open question
- [Jordan] Still some question as to if this is good for the community to do
- [Jordan] Eliminating the disruptive parts of our minor version upgrades is a key point to help people towards smoother use (but all the other things in your cluster are where the risk exists).
- [Mo Khan] Valuing stability above all means only essential changes - and complete testing is key.
- What's next?
    - [Mo] Better upgrade tests and skip-version upgrades
    - [Jordan] Focusing on closing gaps we have today, and leaving EOL branches in a potentially-supportable state. Keep pushing for no-action-required minor-version upgrades. Next up: bring those practices to Kube-adjacent ecosystem projects.
    - [Scott] Reducing the cognitive load of applying an upgrade; limiting issues in ecosystem projects.
- [Jordan] Could we get K8s to the point where the minor upgrade wasn't a problem if all else were held the same? Are we already getting there (no "action-required" items in 1.30, etc). A point to ponder.

April 23, 2024 (recording)

Host: jerickar
Note Taker: TBD
Attendees:

- Add your names here
- Lauri Apple
- Marko Mudrinić
- Bridget Kromhout
- Scott Dodson
- Micah Hausler
- Paco Xu

Recurring Topics [timebox to N min]:

- (if you have any… here are some suggestions)
- Welcome any new members or attendees
- Project board review
- Survey results – I pinged here https://kubernetes.slack.com/archives/CDTMPM75W/p1712673940077399?thread_ts=1709816545.230969&cid=CDTMPM75W and pinged again during the meeting

Open Discussion [timebox to N min]:

- Add your suggested topic here and move this to the line below [firstname lastname, email or @slackname]
    - [Bridget] asked Jordan about go mod update implications - Jordan updates us on "[liggitt] need to follow-up on implications of cri-api bumping go versions on kube master branch if LTS branches of other projects will pick it up"
        - bumping go versions may be something people don't want to do right away, but go is making a "compat" change in 1.23 to make this easier
        - unsure if there's anything K8s should do differently in the meantime, but we have to stay on supported go versions and absorb new runtime changes on new k8s minors
        - As long as an LTS stays within the go team's two-year window, then updates can be applied.
    - [Jeremy] followup on blog posts - Scott interested in one; Jordan interested in discussion with the production readiness team for another.
        - Jordan and Bridget to pick a topic for a blog post
    - [Jeremy] strengthen wording around what can be backported: Strengthen Cherry Pick Guidance · Issue #7634 · kubernetes/community (github.com)

April 9th (recording)

Host: jerickar
Note Taker:
Attendees:

- Brad Ascar - Redis
- Jordan Liggitt - Google
- Vince Power - Dell
- Scott Dodson - Red Hat
- Krzysztof Wilczyński - Red Hat
- Bridget Kromhout - Microsoft

- Mo Khan - Microsoft
- Lauri Apple

Recurring Topics [timebox to N min]:

- (if you have any… here are some suggestions)
- Welcome any new members or attendees

Open Discussion [timebox to N min]:

- Add your suggested topic here and move this to the line below [firstname lastname, email or @slackname]
- [All] KubeCon debrief / thoughts?
    - [micah] contributor summit was kind of a recap of work that has been done and discussion of regression analysis, Go compatibility
    - [Bridget] noted some where do we go from here questions, like “ok what do we do now?” People are looking for things we can’t give them right now [skip upgrades] and things we have but people don’t really know about (things are easier)
        - continued opportunities to document / guide people to use the things that *are* available already to reduce upgrade disruption / frequency (node skip upgrades, etc)
    - [mo] will check and see if contrib summit sessions recordings are up. A lot of conversations around fan out problem of multiple version support.
    - [jordan] that is my biggest concern, people building to lowest common denominator will lag a lot
    - [mo] will transcribe and bring notes for next time. Seems like a topic people care about and there is work to be done. Fairly positive on topic overall.
    - [Jordan] more questions from people responsible for running clusters or people that get forced to versions?
    - [mo] because it was contrib summit, more focused on community / developer side.
    - [mo] further discussion at KubeCon NA would be good
        - [jeremy] I can start a google doc to collaborate on a maintainer track + earned session submission, will share in slack.
    - [mo] action item: bring further analysis of the contrib-summit discussion to this meeting in two weeks’ time
        - Recording: Viability of Kubernetes community LTS
    - [mo] Its model from containerd is willing to backport cri-api-exposure of existing features in previous versions of containerd as cri-api integration solidifies
        - [liggitt] need to follow-up on implications of cri-api bumping go versions on kube master branch if LTS branches of other projects will pick it up
            - bumping `go` directives is viral, go propagates that up to everything that depends on it
- [liggitt] update on regression data
    - Updated the regression-tracking sheet:

        - 1.28 and 1.29 (the minor versions released after my initial sweep and circuit of SIGs early in 2023) are looking way better
        - 1.28 and 1.29 match or beat lowest regression backport counts since 1.19 / 2020
        - 1.28 and 1.29 remain the first minor versions since 2020 that haven't yet regressed in a patch release due to a backport (that we know of)
        -
- [jeremy] Next steps?
    - Blog post highlighting improvements in regressions and release quality
    - Blog post highlighting
    - Survey data and other feedback seems to point to skip upgrades and “easier” upgrades. Any concrete actions we can facilitate in other SIGs to help with this (upgrade testing? Build on compatibility versions?)
        - [jordan] upgrade testing would be a big +1, especially with multi-server clusters. Providers probably do this, but it would be good to shift this left into the community. Easy to say and hard to do. Flakes kept things from being pre-submits, and if it’s not pre-submit any random PR can break it. Things that aren’t pre-submits are really difficult to keep green.
        - [jeremy] good next steps to revisit with cluster-lifecycle and testing?
        - [jordan] yeah, linking into existing convos/reviving them would be good, figuring out how we can help.
        - AI: let’s go talk to sig-testing and sig-cluster-lifecycle
    - [jordan] update / close the backport guidance issue by increasing rigor around backport criteria.
        - https://github.com/kubernetes/community/issues/7634
        - [jordan] gave an example about a CEL bug that didn’t get backported
        - [mo] going back to CEL bug backport that Jordan mentioned, fixed in 1.30 and wanted to backport it and Jordan reminded me that it didn’t fit the criteria for a real backport.
            - (specific example of bug fix that wasn’t backported) https://github.com/kubernetes/kubernetes/pull/123540
        -
    - [lauri] what’s the status of the data analysis? Follow up with Josh re: data dump so we can better analyze
    - [jeremy] follow up to previous items identified from Feb 27th meeting and add to project board so we can follow up with them in a more structured way.

Mar 26

Cancelled for empty agenda

Mar 12 (recording)

Host: liggitt

Note Taker:
Attendees:

- Brad Ascar (Redis)
- Mo Khan (Microsoft)
- Scott Dodson (Red Hat)
- Jordan Liggitt
- Marcin “Perk” Stożek (Canonical)
- Akhil Mohan (VMware)
- Micah Hausler (AWS)

Recurring Topics [timebox to N min]:

- (if you have any… here are some suggestions)
- Welcome any new members or attendees
- Project board review

Open Discussion [timebox to N min]:

- Add your suggested topic here and move this to the line below [firstname lastname, email or @slackname]
- [liggitt] FYI: ecosystem support activity
    - containerd 2.0 / 1.7 support
        - https://github.com/containerd/containerd/pull/9833
        - https://github.com/containerd/containerd/pull/9879
    - Go compatibility retread
        - proposal: cmd/go: separate default GODEBUGs from go language version · Issue #65573 · golang/go · GitHub
- [liggitt] Signal boost from last time
    - Brainstorm: stories of keeping existing things working (or not)
- [mo] I will be at KubeCon EU 2024 contributor summit / SIG meet and greet, any items that I should try to cover/discuss?
    - who will be there?
        - Micah Hausler, Mo
    - Currently have ‘viability of community LTS’ as topic
    - Ask folks maintaining kube releases after OSS EOL what remaining pain points are
    - Take advantage of having maintainers from other projects in the same space (containerd, CRI-O, coredns, OpenTelemetry Operator etc); implications of a given Kubernetes release being used longer
        - Considering the fan-out of add-ons needing to support old versions of Kube
        - Would additional kube OSS versions change or slow use of new features (like inlined custom resource validation rules vs validation webhook) (Gateway project is a good example of this)

Feb 27, 2024 (recording)

Host: Jeremy Rickard

Note Taker: Bridget Kromhout

Attendees:

- Add your names here
- Micah Hausler (AWS)
- Scott Dodson (Red Hat)

Recurring Topics [timebox to N min]:

- (if you have any… here are some suggestions)
- Welcome any new members or attendees
- Project board review

Open Discussion [timebox to N min]:

- Add your suggested topic here and move this to the line below [firstname lastname, email or @slackname]
- Jeremy: Discussion of what we've found this far…?
    - Bridget: people want less disruption (not LTS necessarily, but less terrible upgrades)
    - Micah: slower-to-change teams struggle with being behind on upgrades
    - Jeremy: less-disruptive upgrades and skip-upgrades are what would make the most difference to people
    - Bridget: maybe messaging around "there is hope!"
        - Jordan: people who are stuck are stuck on specific older versions (1.21, 1.25) where upgrading is difficult, "goal is no-action-require upgrades and these are the milestones that have made this better for you"
        - Bridget: do we have a milestone to put this blog post in?
        - Jordan: 1.29 (last default-on beta went GA), or 1.32 (last default-on beta turns off)
        - Jeremy: recommend posting ~now~~, after 1.29 but ahead of 1.30, would be a good time to get this blog post out there~~
        - Jordan: the "no action required" ethos is something we're bringing to our dependencies like etcd, containerd 2.0, etc. - so people don't just have K8s but other stuff is missing
    - Jeremy: people being "stuck" - feature removal, cost/disruption (due to bigger clusters)
    - Scott: could contribute something around reducing workload disruption during upgrades
        - let's work with sig docs/cluster lifecycle/node
        - canary patterns to validate new nodes
        - kubelet skew allowing for multi minor upgrades w/ single workload roll
        - review / improve website documentation / examples
        - signal-boost through blog post
        - Jordan: some solutions exist like canary nodes, don't have to 2x
    - Improving internal mindset to avoid breaking changes
        - building empathy with users
        - categories of breaking changes / action-required changes
            - [making it very clear if it's successful or not]

        - making the principle less abstract and more concrete
            - positive examples: big changes we took a lot of care to roll out non-disruptively that went well where the effort was worth it
            - cautionary examples: changes we knew would be breaking but were more disruptive to users than we expected
        - brainstorm ways to reinforce this for leads / reviewers / approvers / contributors
        - Brainstorm: stories of keeping existing things working (or not)
        - https://groups.google.com/g/kubernetes-sig-architecture/c/j1O3qy1iFI0
    - Update the slide deck with more data summary
        - Send to stakeholder sigs
        - Send to dev@

Feb 13, 2024 (recording)

Host: Micah Hausler
Note Taker: Bridget Kromhout
Attendees:

- Bridget Kromhout (Azure)
- Mo Khan
- Marcin “Perk” Stożek (Canonical)
- Marcin Franczyk (Huawei)
- Scott Dodson (Red Hat)
-
- Add your names here

Recurring Topics [timebox to N min]:

- (if you have any… here are some suggestions)
- Welcome any new members or attendees
    - Perk is a product manager for Kubernetes at Canonical
- Project board review

Open Discussion [timebox to N min]:

- [jeremy] - Some analysis of survey data based on scenarios from last time
    - Scenario 1 (regulatory compliance)
    - Scenario 2 (people stuck)
        - 45% of people have regulatory constraints
        - Only 15 (out of 200) responses for older than k8s 1.21
        - Would be interesting to see how wide a range the typical respondent has (between old and new)
        - Rather than later (which is ambiguous), let’s say “newer” or “older”
        - [Micah] how are regulated teams passing compliance
        - [Bridget] do the scanners actually catch old K8s?
        - [Scott] 1.29 is where the breaking changes on versions get better (so, these people aren’t there yet)

- [Micah] backward-compatibility between kubelet/control plane does help now
- [Bridget] lack of capacity to 2x your estate to upgrade
- [Micah] Risk around updates - time to qualify a new version
- [Micah] There are vendors in this space to help consider compatibility around add-ons for upgrades
- CLI projects to help out with finding deprecated K8s resources
    - Kube No Trouble: https://github.com/doitintl/kube-no-trouble
    - Pluto: https://pluto.docs.fairwinds.com/
- Cost/disruption/removed features (PSP, dockershim, etc)
- [Scott] OpenShift customers keep using until EOL and longer
- [Mo Khan] an LTS that does only security updates will meet the needs (most aren't looking for features)
- [Micah] if a company runs many clusters, they aren't often run by the same team
- [Bridget] Giving people more validation that an upgrade is safe would save them time
- [Bridget] Skip-level seems to be desired
- [Micah] visualizations of data
- [Scott] Interested in how the version-sku upgrades are affecting people (are they using that new capability yet) - OpenShift positions that as an upgrade pattern and customers find it valuable
- [Micah] customers who are doing all the upgrade work decide to do the nodes also as well at the same time.
- Mo: https://github.com/kubernetes-csi/csi-proxy
-

- wg-lts data
-

Jan 30 (recording)

Host: Jeremy Rickard
Note Taker: TBD
Attendees:

- Angelos Kolaitis (Canonical)
- Jordan Liggitt
- Scott Dodson
- Marko Mudrinić
- Paco Xu
- Vince Power (Dell)
- Bridget Kromhout (Microsoft)

- Lauri Apple
- Marcin Franczyk (Huawei)
- YuikoTakada(NEC)
- Krzysztof Wilczyński (Red Hat)
- Arka Saha (VMware by Broadcom)

Recurring Topics [timebox to N min]:

- (if you have any… here are some suggestions)
- Welcome any new members or attendees
- Project board review

Open Discussion [timebox to N min]:

- Start categorizing free text survey responses
    - Initial data cleanup by Lauri Apple to pull out data relevant to sig-release planning around release artifacts, etc
        - copy/transform of survey data as of 12/27/2023
          Charts of Kubernetes Upgrade Survey (Responses)
    - Free-form questions
        - Describe your current upgrade process
        - To what degree is your upgrade process automated?
        - To what degree is your upgrade process automated?
        - Do you have change freeze windows lasting longer than a month where upgrades cannot be applied? If so, how often and how long are those freeze windows?
        - What are your biggest challenges or blockers with upgrading?
            -
        - If a new release was "perfect" (no regressions, no deprecations, no removals, no action-required changes), please share other constraints that would keep you from upgrading, or prevent you from upgrading more regularly.
    - Questions / scenarios
        - Start with:
            - **Do you want to?** What is motivating people to upgrade?
            - **Are you able to?** What is preventing people from upgrading?
            - What solutions exist?
            - Look at **column AB**: What does LTS mean to you? (Intersects with SIG Release's exploratory work into the rel tooling and process)
            - Are they aware of the solutions? (we could make inferences about things like "there's not enough space/capacity")
            - Also: A lot of respondents are on managed, but a lot are not.
        - **Scenario 1:** People with regulatory constraints and see what they're looking for. Jeremy will help.
            - e.g. lag for regulatory / qualification / certification time eats all/most of the support window before a version is even usable

        - concrete question for "special regulatory constraints" mentioned - how much time does each add ahead of consuming a new version?
    - **Scenario 2:** People on old versions getting stuck.
        - are they stuck on things that LTS would help?
        - are they stuck on things that have already been addressed (systemically) in recent versions (like beta versions and API deprecations)?
    - **Scenario 3:** People getting stuck whose cadence is set by themselves vs. cloud-based (getting turned off, but there's nuance as managed services have cadence)
    - **Scenario 4:** users who are happy with the current duration / process
        - themes for what makes them successful?
    - **Scenario 5:** People who want to upgrade but having trouble with it
    - **Scenario 6:** People who don't want to upgrade but trying to avoid pitfalls of breaking things or doing others' bidding (late as possible/never upgrade)
    - **Scenario 7:** Ops who want to upgrade but are blocked by a stakeholder (like an app developer or operator developers)
    - **Scenario 8:** not stuck, but upgrade process / cadence simply cannot keep up?
        - e.g. upgrades take longer than 4 months
        - e.g. cannot accommodate an upgrade every 4 months or a stacked upgrade every year
- Add your suggested topic here and move this to the line below [firstname lastname, email or @slackname]

Jan 16, 2024 (recording)

Host: Jordan Liggitt
Note Taker: Bridget Kromhout
Attendees:

- Add your names here
- Scott Dodson (Red Hat)
- Marcin Franczyk (Huawei)
- Pedro Fragola (Canonical)
- Rita Zhang (Microsoft)
- Arka Saha (VMware by Broadcom)
- Vince Power (Dell)

Recurring Topics [timebox to N min]:

- Welcome any new members or attendees (no new people this time)
- Project board review

Open Discussion [timebox to N min]:

- [liggitt] Compatibility mode KEP opened by @jpbetz
  - Presented a month or two ago in this call - now open as a KEP - relevant to upgrade patterns such as skip-level
  - KEP freeze Feb 8 - the goal is to have this implementable as alpha for 1.30.
  - Please add feedback to the KEP PR
- [liggitt] survey closing … when?
  - Jan 31 - signal boost to mailing list/K8s-dev
- [liggitt] plan for crunching data from the survey
  - volunteer in slack - Lauri Apple, Josh Berkus, Vince Power
  - We will want to bucket the free-form text to try to pull patterns out of these
  - Possibly bucketing-party? Tentatively next meeting (Jan 30)
- [sdodson] kube-linter - Tool to identify workload issues, could be used to help people build workloads compatible w/ upgrades. Open to broader community contrib and/or moving to kube-sigs
  - Some discussion in LTS channel about problems with Workload (Identity?)
  - Give warnings about upgrade-compat or other risks, mostly focused on security and workload health aspects
  - The team for this tool may be interested in moving it into a kube-sig repo. Jordan would like to hear an experience report from them on what they've found interesting to flag.
  - [liggitt] ~~Does it make use of metrics for deprecations / warnings, etc?~~ YAML manifest only analysis
    - maybe that was https://github.com/doitintl/kube-no-trouble
  - Scott has proposed this collab to them - they are interested.
  - Scott will see if they want to present in Feb at this meeting
- [liggitt] Go 1.21 update on release branches queued up for January - first time we've done this update after the automatic backwards-compat so it will _probably_ be smoother
  - https://go.dev/doc/godebug
  - GODEBUG settings added for compatibility will be maintained for a minimum of two years (four Go releases). Some, such as http2client and http2server, will be maintained much longer, even indefinitely.
- AI: identify person to run APAC friendly meeting time, have them create a poll for a meeting time that works for them
-
- Add your suggested topic here and move this to the line below [firstname lastname, email or @slackname]

Dec 19th, 2023

Cancelled

Dec 05, 2023 (recording)

Host: Jordan Liggit
Note Taker: Bridget Kromhout
Attendees:

- Scott Dodson (Red Hat)
- Marcin Franczyk (Huawei)
- Han Kang (Google)
- Arnav Mediratta (AWS)
- Arka Saha (he/him)

Recurring Topics [timebox to N min]:

- (if you have any… here are some suggestions)
- Welcome any new members or attendees
    - Divya (Red Hat)
    - Oliver Stutz (Priverion)
- Project board review
    - Survey is up and open (will be open until early January) - signal boost it!

Open Discussion [timebox to N min]:

- Add your suggested topic here and move this to the line below [firstname lastname, email or @slackname]
    - Jeremy R opened Strengthen Cherry Pick Guidance · Issue #7634 · kubernetes/community · GitHub
    - Scott Dodson asks: do we have a consistent label taxonomy?
    - Jordan: not entirely consistent; it's a bit ad-hoc right now
    - Scott: I will suggest more consistent labels on this issue.
- AI: [jerickar] propose alternating with APAC friendly meeting time
- Han: draft KEP for compatibility-mode open
    - https://github.com/kubernetes/enhancements/compare/master...logicalhan:enhancements:compat-versions
    - Prototypes:
        - https://github.com/kubernetes/kubernetes/compare/master...logicalhan:kubernetes:ff-v1
        - https://github.com/kubernetes/kubernetes/compare/master...logicalhan:kubernetes:ff-v2
        - https://github.com/kubernetes/kubernetes/compare/master...logicalhan:kubernetes:ff-v3
- Front page of the orange site - https://news.ycombinator.com/item?id=38516717
    - [Oliver Stutz] Joined because of HN article - why LTS? Have seen user impact of upgrades decrease over time, not sure why LTS is needed in 1.27+
    - [Jordan] reasons people want vary - we have this survey to discern which reasons may still exist: https://bit./k8s-upgrade-survey
    - Scott: I had put together a document on OpenShift's experiences w/ 24 month alternating duration lifecycle which details some of the downsides like deferred

uptake. linked in the agenda but at
OpenShift's Experiences with 24 months of support
    - Han: if we did LTS in the style of "one blessed LTS version", we need skip-level upgrade to get LTS-to-LTS upgrades; skip-level cluster upgrade isn't supported today
    - Bridget: the purpose of this working group is to explore what should LTS in k8s OSS look like
    - Scott: volunteers to summarize WG meetings/slack activity for mailing list
- Bridget: noticed that a non-zero number of responses were indicating older-than-1.21 - we probably will need a plan for guiding people to replace (not upgrade) far-too-old clusters. (Likely, with blue-green.)
    - Jordan: the people who aren't doing frequent upgrades also will find migration patterns challenging.
    - Oliver: GCP feature showing which APIs you're using is great to help encourage upgrades - we should guide people using K8s as to how to safely upgrade (reduce their fear). (The ecosystem expects flexibility - we should continue that. LTS isn't for everyone.)
    - Jordan: The community may _not_ do an LTS version - we're identifying changes that will help everyone (ie, upgrades more reliable). And for action-required, we want it to be possible to programmatically pre-flight their upgrades (instead of manually checking changelogs). Upshot: life is better for everyone.
    - Oliver: Consider costs of re-certification (perhaps version-numbers would differ such as to indicate that the change is much smaller) - it should be possible to patch asap
    - Jordan: K8s changes how orgs run production systems; we need to balance reality with ideal outcomes
    - Han: the aspiration is to be semver compliant for rest apis, config file apis, etc
    - Jordan: 1.32 will be the first semver-compliant version (since 1.24 no new unstable APIs on by default)
        - *[lubomir/neolit123] out-of-band clarification: as per the meeting VOD, 1.32 can be SemVer compliant in terms of REST APIs.*
    - Han: you won't need deprecated-api-detectors when semver-compliant
    - Jordan: most of the HN issues were due to assuming beta APIs were stable (they weren't) - once the ecosystem gets past 1.21->1.22 these factors will change (current versions won't have these problems)
- Han: youtube meeting uploads have to be done manually

Nov 21, 2023 (recording)

Host: Jeremy Rickard (Microsoft)
Note Taker: Bridget Kromhout (Microsoft)
Attendees:

- Jordan Liggit (Google)
- Scott Dodson (Red Hat)

- Marcin Franczyk (Huawei)
- Arka Saha(he/him, VMware)
- Pedro Fragola (Canonical)
- Rita Zhang (Microsoft)
- Krzysztof Wilczyński (Red Hat)
- Lubomir I. Ivanov (VMware)
- Yash Singh (VMware)

Recurring Topics [timebox to N min]:

- (if you have any… here are some suggestions)
- Welcome any new members or attendees
    - Marcin Franczyk - joining Joe to represent Huawei
- Project board review
    - Nothing on the project board this time

Open Discussion [timebox to N min]:

- Add your suggested topic here and move this to the line below [firstname lastname, email or @slackname]
- [liggitt] review regressions data compiled for 1.19-1.28 (if time permits)
    - Kubernetes patch release regression/bugfix rate
    - Analysis of Kubernetes regression rates, patterns, examples
    - (docs are shared with https://groups.google.com/g/kubernetes-wg-lts and https://groups.google.com/a/kubernetes.io/g/dev)
        - Jordan has gone over this data with a variety of audiences; in this case he's highlighting what's specifically interesting for LTS
        - Starting with 1.19 when we moved to annual release cycles
        - Most regressions were present in a .0 release, but occasionally patch releases and subsequent backports broke things (which is of interest for LTS)
        - Client-side regressions sometimes come to light when server-side changes land (like between 1.26 and 1.27).
        - Every single minor version has had a backport cause a regression. This is the thing we're the most worried about for LTS backports.
        - Backports affecting seemingly-far-off code areas or out-of-tree users are harder to notice regressions in (until they hit).
        - Jordan talked with sig release about how to maintain this data going forward (labels, etc)
        - New fixes to an old code branch to fix a CVE when we cannot backport: this requires us to rewrite a fix or write a novel fix.
        - Lack of test coverage for an e2e scenario can expose issues
        - These issues are the ones not caught by CI - but note that bugfixes can be riskier to backport than features (in terms of regressions surfacing).
        - Discussion surfaces the fact that a number of teams are starting with only CVE-fix backports, not bugfix backports
        - Discussion about filtering out false positives.

        - Suggested bar for backporting bugfixes: regression fixes, security fixes, data loss fixes. For anything else, more justification should be required.
        - Defect classification and cherry-pick justification: concrete steps we could take now.
            - limit backports to regression / security / data-loss fixes by default
            - anything beyond that needs justification / careful risk analysis (size, entanglement with other changes, test coverage)
            - the level of scrutiny changes to master get between code freeze and .0 release is the level we should be applying to backports
                - @jeremyrickard logged a ticket:
                    - Strengthen Cherry Pick Guidance · Issue #7634 · kubernetes/community (github.com)
- [jerickar] review initial survey results
    - Some regulatory/compliance needs (perhaps a quarter of responses)
    - Jeremy to analyze free-form text responses before the next meeting
    - Many of the concerns around upgrades may be less of an issue now due to improvements around deprecation handling
    - Users want at least two years of support and skip-upgrades (the latter has complexities)
    - Storage and networking along with capacity can be blockers to prevent cluster replacement.
- [jerickar] propose alternating with APAC friendly meeting time

Nov 7, 2023 - no meeting (KubeCon)

Oct 24, 2023 (recording)

Host: Jeremy Rickard (Microsoft)
Note Taker: Bridget Kromhout (Microsoft)
Attendees:

- Xander Grzywinski (he/him, Microsoft)
- Angelos Kolaitis (he/him, Canonical)
- Pedro Fragola(Canonical)
- Scott Dodson (Red Hat)
- Paco Xu(he/him, DaoCloud)
- Rita Zhang (Microsoft)
- Vipin Mohan (AWS)
- Arnav Mediratta (AWS)
- Joe Huang(Huawei)
- Konstantinos Tsakalozos (he/him, Canonical)
- Marko Mudrinić (he/him, Kubermatic)

Recurring Topics [timebox to N min]:

- (if you have any… here are some suggestions)

- Welcome any new members or attendees
    - Vipin Mohan - product manager at AWS
- Project board review

Open Discussion [timebox to N min]:

- [jerickar] review survey as a group before we publish it, ahead of KubeCon
    - https://docs.google.com/forms/d/e/1FAIpQLSfixO9RHf8GMh9bgS7RE6NcbCYM1yzvWINSF0axozmQSAWhtw/viewform?usp=sharing
    - Some suggestions in this doc, below
    - Let's work on this sync - Jeremy sharing WIP survey
        - Adding clarity to questions and refining the order/whether they're mandatory
    - **Review / edits in by next Monday, Oct 30 (thread on Slack)**
    - **Target publish next Tuesday, Oct 31**

Oct 10th, 2023 (recording)

Host: Jeremy Rickard (MSFT)
Note Taker: Rita Zhang
Attendees:

- Marko Mudrinić
- Xander Grzywinski (Microsoft)
- Krzysztof Wilczyński (Red Hat)
- Scott Dodson (Red Hat)
- Konstantinos Tsakalozos (Canonical)
- Caroline Davis (SUSE - Rancher)
- Jordan Liggitt (Google)
- Angelos Kolaitis (Canonical)
- Joe Huang(Huawei)
- Rita Zhang (Microsoft)
- Humble Chirammal(VMware)
- Antonio Ojea (Google)
- Arka Saha (VMware)
- Nigel Brown (Intuit)
- Paco Xu(DaoCloud)
- Tim Pepper (VMware)
- Micah Hausler (AWS)

Recurring Topics [timebox to N min]:

- (if you have any… here are some suggestions)
- Welcome any new members or attendees
- Project board review

Open Discussion [timebox to N min]:

- Add your suggested topic here and move this to the line below [firstname lastname, email or @slackname]
- [Scott Dodson] Observations from Red Hat's non-EUS/EUS 18/24m lifecycle
    - About 24 months ago introduced EUS lifecycle where Odd Kube versions receive 24 months and embraced kubelet version skew to promote "EUS-to-EUS" upgrades w/ single workload disruption, predates recent change in 1.28 to support N-3 to N.
    - OpenShift's Experiences with 24 months of support
    - Do you backport features for EUS?
        - We try hard not to but it depends on customer asks. we generally have few feature backports
    - How do components running on top of k8s get updated/shipped for EUS?
        - we asked ISVs to support their components for 24mons and sometimes they do need to be bumped once or twice
        - as long as they support the oldest version and newest version
        -
    - What type of policy do you have around reporting and patching CVEs for EOL k8s?
        - we pick up fixes RHEL provides
        - we eval new CVEs for older versions, e.g. Go, k8s CVEs
        - I'm sure we have gotten CVEs for older versions that no longer reported to upstream
    - Are there version bumps to fix CVEs of dependent components like etcd, coredns… for a EOL k8s version?
        - yes we have had to bump to pick up fixes, generally try not to bump minor versions
    - update Go version in k8s or patch Go
        - we have been patching Go
    - how do you deal with CVE in older version but not qualified for newer k8s version e.g. https://github.com/kubernetes-sigs/metrics-server/issues/1289
        - openshift is monolithic, many components are baked in
        - [Krzysztof] We would carry a downstream patch if needed, for as long as the version in a particular release is supported.
        - Downstream patches get reviewed and validated with their own tests
        - [liggitt] for k8s upstream, we do not touch vendor dependencies to ensure they are the same
    - takeaways:
        - non-EUS has seen a significant decline
    - [Scott] For a community project, having an alternating lifecycle can be problematic
- [Jeremy] Started a survey in Google Forms Based on WG-LTS Survey Topics
    - https://docs.google.com/forms/d/e/1FAIpQLSfixO9RHf8GMh9bgS7RE6NcbCYM1yzvWINSF0axozmQSAWhtw/viewform?usp=sharing

- Please review and make suggestions so we can review it together in the next WG meeting
- Ideally sent out BEFORE kubecon Chicago
- Related, PRR survey results: https://github.com/kubernetes/community/blob/master/sig-architecture/Kubernetes_PRR_Survey_2023Q2.pdf (most recent, was done in May so 1.24 was the supported version at that point)
- https://github.com/kubernetes/community/blob/master/sig-architecture/Kubernetes_PRR_Survey_2022Q2.pdf
- https://github.com/kubernetes/community/blob/master/sig-architecture/Kubernetes_PRR_Survey_2021Q2.pdf
- https://github.com/kubernetes/community/blob/master/sig-architecture/Kubernetes_PRR_Survey_2020Q1.pdf
-

Suggestions for the Google form:

1. What is an ideal upgrade cadence for you?
    a. Options:
        i. One upgrade per year
        ii. Two upgrades per Year
        iii. One upgrade every 2 years
        iv. Other
    b. Why is that cadence ideal for you?
2. What part of the upgrade process is the most complex? Most time consuming?
    a. Options:
        i. Preparing for the upgrade (reading changelogs, qualification process)
        ii. Performing the upgrade itself
        iii. Post-upgrade checks and validation
        iv. Other
3. What does an LTS Kubernetes mean for you?
    a. Options:
        i. Security patching for Kubernetes components for the duration of LTS
        ii. Backporting bug fixes and making them available in the LTS Kubernetes version
        iii. Backporting new features and making them available in the LTS Kubernetes version
        iv. Other (please explain)

Sept 25th, 2023 cancelled

Host: TBD
Note Taker: TBD
Attendees:

- Add your names here

Recurring Topics [timebox to N min]:

- (if you have any… here are some suggestions)
- Welcome any new members or attendees
- Project board review

Open Discussion [timebox to N min]:

- Add your suggested topic here and move this to the line below [firstname lastname, email or @slackname]

Sept 12th, 2023 recording

Host: Jeremy Rickard
Note Taker: Jordan Liggitt
Attendees:

- Jordan Liggitt
- Nick Young
- Scott Dodson (Red Hat)
- Marko Mudrinić
- Tim Pepper
- Rita Zhang
- Joe Huang
- Bridget Kromhout (Microsoft)
- Akhil Mohan
- Konstantinos Tsakalozos
- Angelos Kolaitis
- Krzysztof Wilczyński (Red Hat)
- Bhawana Haritwal
- Humble Chirammal (VMware)
- Caroline Davis (SUSE - RKE2/K3s)

Recurring Topics [timebox to N min]:

- (if you have any… here are some suggestions)
- Welcome any new members or attendees
    - Konstantinos Tsakalozos - Canonical
    - Paco Xu - DaoCloud
    - Marko Mudrinic - interested from sig-release and
    - Krzysztof Wilczynski
    - Tim Pepper - vmware, prior lts wg involvement
    - Ben Elder - google, lots of project areas (testing, infra,
- Project board review

FYI

- [Public] Safer Kubernetes Upgrades discussed in sig-arch (9/7) and sig-api-machinery (9/6), initial steps got good feedback, next step is talking with cluster-lifecycle and starting formal KEP, targeting reviewable by kubecon

- [liggitt] analysis of regression rates / areas since 1.19 (preview of topics for sig-arch and sig-release later) - Kubernetes patch release regression/bugfix rate

Open Discussion [timebox to N min]:

- [jeremy] review survey questions doc WG-LTS Survey Topics
    - Lots of ideas logged into the doc ^^
    - Aim for ready to share at KubeCon?
    - If possible, structure to require ranking, not just +1'ing each "would you like __ better?"
- [jeremy] what blockers exist assuming we agreed to provide LTS releases
    -

August 29th, 2023 recording

Host: Jeremy Rickard
Note Taker: Bridget Kromhout
Attendees:

- Jordan Liggitt
- Scott Dodson
- Nigel Brown
- Bridget Kromhout
- Nick Young
- Angelos Kolaitis
- Rita Zhang
- Noah Abrahams
- Humble Chirammal
- Akhil Mohan
- Arka Saha
- Ciprian Hacman
- Anusha Ramineni
- Joe Huang
- Bhawana Haritwal

Recurring Topics [timebox to N min]:

- Welcome any new members or attendees
    - Nigel Brown, Intuit - active in SIG Contribex - developer advocate
    - Akhil Mohan - VMware - SIG Node / containerd
    - Arka Saha - VMware - LTS K8s releases within vmware - test infra
    - Humble Chirammal - SIG Storage, K8s @ VMware
- Project board review
    - Jordan points out it would be helpful to add your name to any project board items and friction log entries for follow-up questions / clarifications

Open Discussion [timebox to N min]:

- [logicalhan, jpbetz, liggitt] - [Public] Safer Kubernetes Upgrades
    - Current assumptions around beta APIs, feature flags, etc assume that upgrades are N+1

        - This doc outlines a different (safer) upgrade process - “compatibility mode” as a precursor to activation.
        - Tactical steps to enable could include formalizing the currently-in-a-table versions for alpha-beta-stable on features.
        - Already talked to sig-arch; talking to LTS now; planning to talk to SIG cluster lifecycle before starting the KEP.
        - Conformance tests - opt-in
        - Deprecation considerations
        - Would need to revisit all policies around feature deprecation to cover skip-upgrade periods of time (number of versions, version skew, timeline for removal)
        - [logicalhan] “My plan is to work on getting consensus first, then draft a KEP by Kubecon, with alpha in 1.30”
- [jeremy] Upgrade Friction/Experience Log - Kubernetes Upgrade Experience Friction Log
    - API removals tend to make a release harder to consume
    - Let’s review next meeting and determine if we want to share more broadly - will want a longer time on this one
    - [Jordan] Interested in either things we haven’t addressed, or things we thought we’d addressed that are still a problem
    - [Jeremy] CTA for the experience log - asking people if they can come share at the meeting
- [jeremy] WG-LTS Survey Topics
    - Work on this over the next two weeks and discuss at next meeting

August 15th, 2023 recording

Host: Jeremy Rickard
Note Taker: Jordan Liggitt
Attendees:

- Jeremy Rickard
- Jordan Liggitt
- Nick Young
- Han Kang
- Micah Hausler
- Bridget Kromhout
- Rita Zhang
- Scott Dodson
- Noah Abrahams
- Joe Huang
- Josh Duffney
- Angelos Kolaitis
- Vince Prignano
- Nikola Prokopic

- Vince Power
- Bhawana Haritwal
- Joe Huang (Chaoyi Huang)
- Shiming Zhang
- Dan Budris
- Paco Xu

Recurring Topics [timebox to N min]:

- (if you have any… here are some suggestions)
- Welcome any new members or attendees

Open Discussion [timebox to N min]:

- Intros
    - Jeremy - sig-release, azure
    - Jordan - architecture/api, google
    - Micah - security release committee, aws
    - Bridget - sig-cloud-provider, azure
    - Vince Prignano - sig-cluster-lifecycle, red hat, end user happiness, easier upgrades, longer cycles
    - Vince Power - dell, end user, very invested in sustainability
    - Arnav Mediratta - AWS
    - Nick Young - Isovalent
    - Noah Abrahams - TPM at oracle, past experience with regulated industries
    - Rita Zhang - sig-auth/security response, azure, focused on making life better for end users
    - Han Kang - understand mechanisms of LTS, google
    - Josh Duffney - end user, at microsoft
    - Scott Dodson - Red Hat, work on OpenShift lifecycle and upgrades
    - Bhawana Haritwal - end user, worked on 1.25 release team, interested in how to keep up with kubernetes releases better
    - Joe Huang - end user, Huawei
    - Caroline - project manager for K3s/RKE2 distros, SUSE
    - Nikola - Giant Swarm (kubernetes as a service provider),
    - Jyoti Mahapatra - EKS, work on cluster provisioning
    - Angelos Kolaitis - engineer for microk8s distribution, canonical. kubernetes provider, have customers asking for LTS, happy to contribute to WG
    - Peter Rifel - datadog, end user, large stateful workloads making upgrades harder
    - Dan Budris - aws EKS Distro
    - Shiming - DaoCloud, have customers they need to provide security fixes to
    - Paco - DaoCloud, have customers they need to provide security fixes to
    - Verónica López- sig-release & PlanetScale
- [Jeremy Rickard] Initial scope of work
    - https://github.com/kubernetes/community/blob/master/wg-lts/charter.md#scope
    - “first phase of the working group, we will collect information related to the needs and wants regarding support periods from end-users and people supporting Kubernetes”

- Noah: important to remember there are people who are not current Kubernetes users *because* of the existing cycle / support posture
- Micah: useful to review / state clearly what the current support timelines are
    - https://kubernetes.io/releases/
    - https://kubernetes.io/releases/patch-releases/
    - OSS supports a minor version for ~14 months (usually rounds up to 15 months)
    - OSS releases 3 new minor versions a year
    - OSS releases a patch release on each minor version ~monthly
    - Jordan:
        - Supported skew between control planes and nodes is n-3 (as of 1.25 nodes / 1.28 control planes)
        - Designed to support an annual upgrade cycle
        - Nodes *can* do skip-version upgrades
        - Control planes *cannot* do skip-version upgrades
    - Rita:
        - Kubernetes Security Response Committee only fields reports and manages fixes for minor versions within OSS supported
- Jeremy: is it valuable to gather data on existing state from existing users
    - Jordan: to avoid overtraining on the past, would be useful to know what things are still an issue for users who have made it to versions that improved known problems (1.23+ on supported go versions, 1.24+ avoiding enabling new beta APIs by default)
    - Micah: the biggest thing we've seen help is no longer enabling unstable beta APIs by default
    - Nick: quick way to start would be a google doc gathering questions we want to ask
- Bhawana: can we get a list of changes / improvements that *have* been made here?
    - [quick braindump from Jordan]
    - 2019: Kubernetes 1.17+ releases supported for 12+ months
        - KEP-1498
        - Cadence changed from 4 → 3 releases each year
        - Support period lengthened from 9 months → ~14 months
        - Enables annual upgrade cycles / use of a minor version for a year while staying on a minor version that gets patch releases
    - 2020: Kubernetes 1.19+ ensures all APIs required to run clusters are GA
        - KEP-1333
        - New APIs are not allowed to be required until they graduate to GA
    - 2020: Kubernetes 1.19+ returns warnings, records metrics and audit annotations when deprecated APIs are used
        - KEP-1693
        - https://kubernetes.io/blog/2020/09/03/warnings/#metrics

        - Cluster admins can use metrics / audit logs as a pre-upgrade guard to avoid upgrading clusters where about-to-be-removed APIs are actively being used
    - 2020: Kubernetes 1.19+ releases require "production readiness" reviews for changes included in release
        - KEP-1194
        - Requires information about upgrade, compatibility, whether the change modifies default behavior, monitoring, etc
    - 2022: Deprecation policy updated to make stable API versions permanent
        - https://github.com/kubernetes/website/pull/31389
        - Applications/clients that use stable APIs will continue to have access to those APIs for all future Kubernetes 1.x versions
        - (Kubernetes had never dropped a stable API version, but this change made that an explicit policy)
    - 2022: Kubernetes 1.24+ only enables new APIs by default once stable
        - KEP-3136
        - New unstable APIs (beta APIs) are off by default
        - Existing on-by-default beta APIs prior to 1.24 remain until their deprecation/removal, but that number is shrinking dramatically and is currently limited to the flowcontrol beta APIs
            - 1.16: 12 beta APIs removed, 28 beta APIs on by default
            - 1.22: 23 beta APIs removed, 11 beta APIs on by default
            - 1.25: 7 beta APIs removed, 6 beta APIs on by default
            - 1.26: 3 beta APIs removed, 5 beta APIs on by default
            - 1.27: 1 beta API removed, 4 beta APIs on by default
            - 1.29: 2 beta APIs removed, 2 beta APIs on by default
            - 1.32: 2 beta APIs removed, 0 beta APIs on by default
    - 2023: Kubernetes 1.23+ release branches stay on supported go versions
        - Since January 2023, all Kubernetes 1.23+ patch releases are on supported Go versions with latest Go security fixes
        - KEP-3744
        - Blog post
    - 2023: Kubernetes 1.28+ control planes support n-3 nodes
        - KEP-3935
        - Updated skew policy
        - 1.25 nodes / 1.28 control planes are supported
        - Allows annual node upgrades with a single node version upgrade (nodes can do skip-level upgrades)
- Bridget:
    - have seen people stuck behind on older versions get confused about what their actual risks are for doing catch-up upgrades
    - in addition to docs/lists, tools or mechanisms to detect if the changes apply to them

- Jordan: for some things (like use of beta APIs), there are metrics/audit logs that make it possible to see if there are clients relying on those prior to upgrade, but use is left as an exercise for the cluster administrator
- Vince Power:
    - Tools like pluto to check for cluster compatibility
- Jyoti:
    - Certification for certain industries, every new version has to go thru certifications
    - Jordan: that might be where node level skip version upgrade might be useful; maybe only control plane need to be certified
    - Joe: Certification is usually based on Kubernetes distribution granularity, not on part of the product for example only the control plane part. If each minor version of control plane needs to be certified, then 1.28 control plane + 1.25 nodes, 1.28 control plane + 1.26 nodes, 1.28 control plane + 1.27 nodes, 1.28 control plane + 1.28 nodes distribution have to be delivered to the customer for certification. The effort for integration, test, certification for all components, container app compatibility test in each distribution is almost the same as releasing each minor kubernetes distribution. Each distribution has to be upgraded in hundreds of sites with independent k8s instances, then it'll make the customer work on the upgrade every day(to avoid risk, won't upgrade many sites in parallel). It leads to lots of complaints, and is not acceptable, and mostly done on major version upgrades, skipping some minor versions both for control plane and node version.
- Rita: Hoping this WG can help many CNCF project define their own LTS
    - Angelos Kolaitis: At the very least serve as a common target version for CNCF projects to support that
- Micah - Can only safely roll back one version, once you apply a subsequent update you cannot go back to the original version.
    - Is rollback tested? Unsure, tested at unit and integration levels.
- Rita: Sounds like we need to improve this https://kubernetes.io/docs/tasks/administer-cluster/cluster-upgrade/ to add all the supported/tested cases
- Jordan
    - understanding current state of users/vendors
    - documenting current state of what's possible with skip-level offline upgrades (even if we only document it "internally" and don't put it on the website yet if there are too many sharp edges)
        - The API server is able to read and handle data persisted in etcd by older releases (all the way back to 1.0)
        - Newer API servers currently only guarantee the data they persist can be preserved and considered valid by an n-1 API server
            - apiVersion of objects persisted in etcd (any time an object is written)

        - relaxed validation of data in fields (if a user makes an update that makes use of the relaxed validation)
        - data in new fields
    - If an API server at version N starts and writes data to etcd, it is not guaranteed safe to start an API server older than N-1 running against that etcd data. A rollback to older than N-1 would require an etcd data restore from prior to any writes via the version N server to be guaranteed safe (which has the possibility of dropping objects created / updated since the etcd backup)
    -
- [Jeremy Rickard] How do we want to track work for the working group?
    - Github Project Board? We could review this at the WG meetings to keep things moving forward
    -