Growing in Your K8s Contributor Role

Tim Pepper <tpepper@vmware.com>
@pythomit tpepper

vmware

Who am I

K8s newbie

- Got involved in 2017
- Org member since early 2018
- 1.10 release team “issue triage” shadow
- 1.11 release team “issue triage” lead

But...

Experienced software engineer

- 20yrs experience, basically all open source dev
- VMware OSPO, Intel OTC, IBM LTC
- Drivers, kernel, distro, s/w update, storage, security, HPC, cloud orchestration

I’ve made a career in open source systems software and have been around this block a few times...learning new technologies, learning new-to-me open source community, participating. I got involved in kubernetes in 2017 as a recognition that a project I’d been working on since 2015 (a golang based orchestrator of VMs and containers) wasn’t going to have a future...the orchestration software ecosystem is consolidating.

I was following k8s activities in a number of specific technology area SIGs, as well as SIG-Docs and SIG-Contribex because I saw needs in those areas of the bigger software engineering and software lifecycle story.

In Contrib-Ex there was a call for folks to help out with “carrying water and chopping wood” on the release process. I’ve done lots of releases on lots of projects and products so figured I was being called...and I volunteered.

Software Engineering & K8s

This is a talk about software engineering. In the context of kubernetes.

There's a giant splat of kubernetes and software engineering words.

Software Engineering & K8s

Software engineering is much more than just writing code

Only one of the words is “programming”. Software engineering is about a whole lot more than just writing code. If you’re worried that you don’t have enough experience in AAA or BBB, don’t worry. None of us knows everything about everything. That’s why we’re having this talk. To explore areas of growth.

My career experiences have really taught me that there are many aspects to doing successful software engineering and it is all technical.

Each of these words could be an area in which you know something or not...an area in which you want to know more.

Either presents opportunity.

Opportunity to learn and grow in areas you don’t yet have all the experience you want or need.

Opportunity to lead in areas where you have experience.

Never do a poll during a presentation

New to K8s

Looking to grow in K8s community

New to software development

Looking to grow in software development career

Who’s who today?

Show of hands, and you can raise your hand multiple times.

Never do a poll during a presentation

New to K8s

Looking to grow in K8s community

New to software development

Looking to grow in software development career

Through the presentation I'll sprinkle these little icons next to relevant topics

Org Member?

https://git.k8s.io/community/community-membership.md

Role	Responsibilities	Requirements	Defined by
member	active contributor in the community	sponsored by 2 reviewers. multiple contributions to the project.	Kubernetes GitHub org member.
reviewer	review contributions from other members	history of review and authorship in a subproject	OWNERS file reviewer entry.
approver	approve accepting contributions	highly experienced and active reviewer + contributor to a subproject	OWNERS file approver entry
subproject owner	set direction and priorities for a subproject	demonstrated responsibility and excellent technical judgement for the subproject	sigs.yaml subproject OWNERS file *owners* entry

I mentioned I became a member this year.

What constitutes org membership is a Frequently Asked Question.

The first step up from being a casual or new contributor to k8s is becoming a member. This means you're part of the kubernetes "org" on GitHub. This gives you ability to add some basic labels to issues and gets your PRs automatically run through "CI" or continuous integration testing. And it gives you responsibility to do what you can within your abilities.

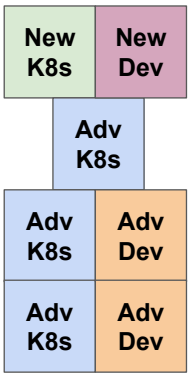
The bar for membership is fairly low. The url on this slide goes into the details. Basically it amounts to demonstrating sustained commitment to the project. You are an "active" contributor.

Where "member" is about being "in the door", the next three levels of "reviewer", "approver" and "owner" are about exhibiting ownership over code. Open source projects achieve the best stability and maintenance over time when individuals feel a personal sense of ownership over areas of code. The opposite is "the tragedy of the commons", the idea that if shared things...resources, code...are owned by nobody, then nobody will care for them, and they will eventually fall into disrepair.

Org Member?

https://git.k8s.io/community/community-membership.md

Role	Responsibilities	Requirements
member	active contributor in the community	sponsored by 2 reviewers. multiple contributions to the project.
reviewer	review contributions from other members	history of review and authorship in a subproject
approver	approve accepting contributions	highly experienced and active reviewer + contributor to a subproject
subproject owner	set direction and priorities for a subproject	demonstrated responsibility and excellent technical judgement for the subproject

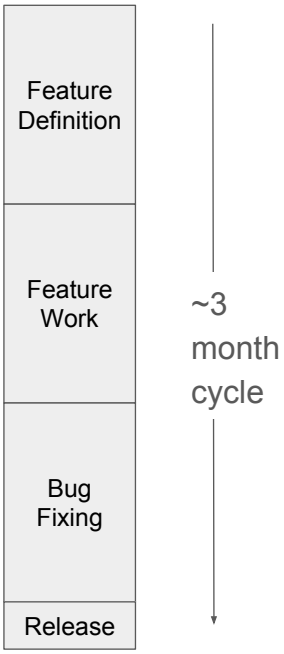


The Release Process

- Feature Discussion (per SIG): ongoing
- Feature Freeze: week ~4
- Release Branch Creation: week ~7
- Code Slush / Freeze: week ~8

...bugs, testing, bugs, fixing, bugs, ...iterating

- End Code Freeze: week ~12
- Release: week ~13



In modern era of agile software development, open source, and git...there really is discussion and development happening all the time.

But at some point you start slowing a bit and peeling off a stable release branch.

This is just normal software engineering. But if that's familiar, it can be a jumping off point to understanding a LARGE open source project like kubernetes

Release Team (1.10)

Role	Name (GitHub/Slack ID)	Shadow Name(s) (GitHub/Slack ID), ...
Lead	Jaice Singer DuMars (@jdumars)	
Features	Ihor Dvoretskyi (@idvoretskyi)	
CI Signal	Sen Lu (@krzyzacy)	
Test Infra	Cole Wagner (@cjwagner)	Benjamin Elder (@BenTheElder)
Bug Triage	Josh Berkus (@jberkus)	Tim Pepper (@tpepper)
Branch Manager	Caleb Miles (@calebamiles)	
Docs	Jennifer Rondeau (@Bradamant3)	
Release Notes	Nick Chase (@nickchase)	Noah Abrahams (@qedrakmar)
Communications	Natasha Woods (@nwoods)	Kaitlyn Barnard / Caitlyn O'Connell
Patch Release Manager	Maciek Pytel (@MaciekPytel)	

Ideally an area shadow role is preparing you to be an area lead in the next release.

Release Team (1.11)

Role	Name (GitHub/Slack ID)	Shadow Name(s) (GitHub/Slack ID), ...
Lead	Josh Berkus (@jberkus)	
Features	Ihor Dvoretskyi (@idvoretskyi)	Stephen Augustus (@justaugustus)
CI Signal	Aishwarya Sundar (@AishSundar)	
Test Infra	Benjamin Elder (@BenTheElder)	Cole Wagner (@cjwagner/cjwagner)
Bug Triage	Tim Pepper (@tpepper)	Cole Mickens (@colemickens)
Branch Manager	Caleb Miles (@calebamiles)	Sen Lu (@krzyzacy)
Docs	Zach Corleissen (@zacharysarah)	Zach Arnold (@zparnold), Misty Stanley-Jones (@mistyhacks), Tom van Waardhuizen (@tallt0m)
Release Notes	Nick Chase (@nickchase)	Mike Arpaia (@marpaia)
Communications	Kaitlyn Barnard (@kbarnard10)	Andrew Hatfield (@andrewhatfield)
Patch Release Manager	Anirudh Ramanathan	@foxish

Ben Elder, Tim Pepper, Kaitlyn Barnard moved up from shadow to lead.

Josh Berkus moved up from bug triage to overall release lead.

Having multiple shadows can be a way to prepare the pipeline for the future. Sharing knowledge, spreading knowledge.

Across these roles there is room for folks from all experience levels and opportunity to build skills.

Release Team (1.11)

Role	Name (GitHub/Slack ID)	Shadow Name(s) (GitHub/Slack ID), ...
Lead	Josh Berkus (@jberkus)	
Features	Ihor Dvoretskyi (@idvoretskyi)	Stephen Augustus (@justaugustus)
CI Signal	Aishwarya Sundar (@AishSundar)	
Test Infra	Benjamin Elder (@BenTheElder)	Cole Wagner (@cjwagner/cjwagner)
Bug Triage	Tim Pepper (@tpepper)	Cole Mickens (@colemickens)
Branch Manager	Caleb Miles (@calebamiles)	Sen Lu (@krzyzacy)
Docs	Zach Corleissen (@zacharysarah)	Zach Arnold (@zparnold), Misty Stanley-Jones (@mistyhacks), Tom van Waardhuizen (@tallt0m)
Release Notes	Nick Chase (@nickchase)	Mike Arpaia (@marpaia)
Communications	Kaitlyn Barnard (@kbarnard10)	Andrew Hatfield (@andrewhatfield)
Patch Release Manager	Anirudh Ramanathan	@foxish

Adv K8s | Adv Dev

Adv K8s | New K8s

New K8s

New K8s

Adv Dev

Adv K8s | Adv Dev

Adv K8s | Adv Dev

Ben Elder, Tim Pepper, Kaitlyn Barnard moved up from shadow to lead.

Josh Berkus moved up from bug triage to overall release lead.

Having multiple shadows can be a way to prepare the pipeline for the future. Sharing knowledge, spreading knowledge.

Across these roles there is room for folks from all experience levels and opportunity to build skills.

Release Team (1.11)

Role	Name (GitHub/Slack ID)	Shadow Name(s) (GitHub/Slack ID), ...
Lead	Josh Berkus (@jberkus)	
Features	Ihor Dvoretskyi (@idvoretskyi)	Stephen Augustus (@justaugustus)
CI Signal	Aishwarya Sundar (@AishSundar)	
Test Infra	Benjamin Elder (@BenTheElder)	Cole Wagner (@cjwagner/cjwagner)
Bug Triage	Tim Pepper (@tpepper)	Cole Mickens (@colemickens)
Branch Manager	Caleb Miles (@calebamiles)	Sen Lu (@krzyzacy)
Docs	Zach Corleissen (@zacharysara	Zach Arnold (@zparnold), isty Stanley-Jones (@mistyhacks), m van Waardhuizen (@tallt0m)
Release Notes	Nick Chase (@nickchase)	Mike Arpaia (@marpaia)
Communications	Kaitlyn Barnard (@kbarnard10)	Andrew Hatfield (@andrewhatfield)
Patch Release Manager	Anirudh Ramanathan	@foxish

Adv K8s | Adv Dev

Adv K8s | New K8s

New Dev

New Dev

New Dev

Adv K8s | Adv Dev

Adv K8s | Adv Dev

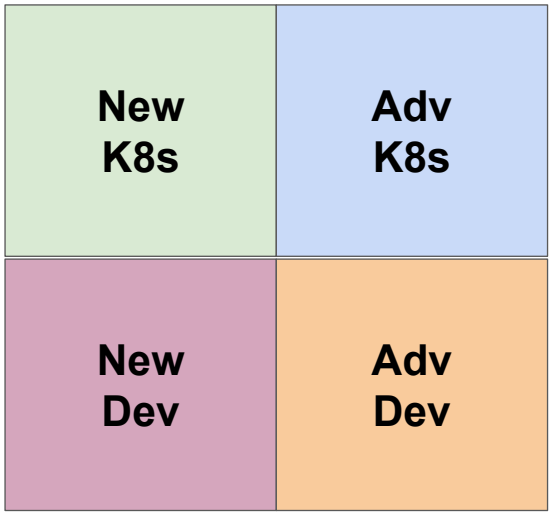
New K8s

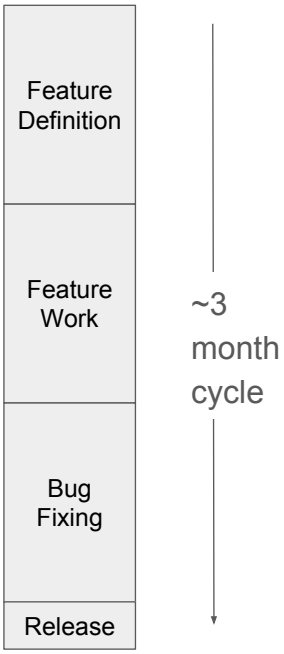
New K8s

Adv Dev

Even those new to software development: test failures and bug reports can be a jumping off point to understand something. You may not be the person who ultimately resolves the issue, but try to run the test, try to understand how it interacts with the code, as you gain understand is there documentation you can improve?

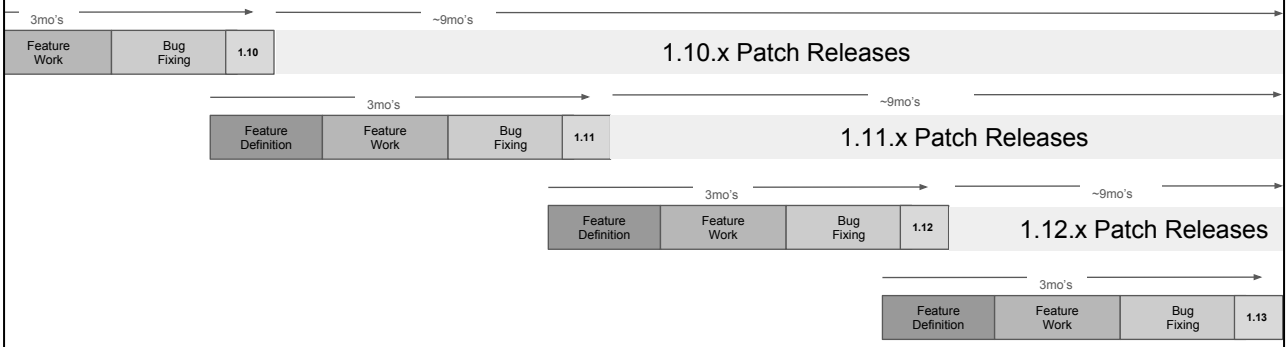
The **Volunteer** Process





If you see the release process as a jumping off point to learn about new areas of k8s, then volunteer.

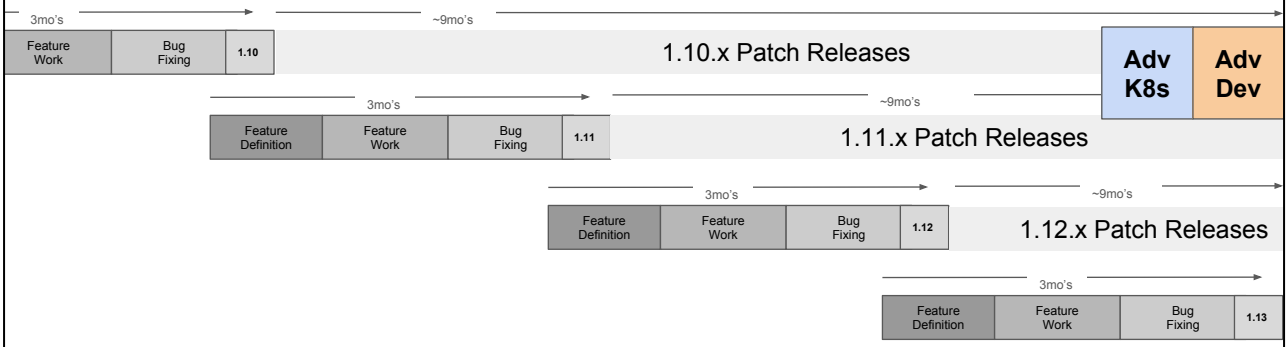
The **Volunteer** Process



Rinse and repeat

The release process needs a steady stream of volunteers. Even if you're not sure about it now...every three months a new cycle is starting...

The **Volunteer** Process



Rinse and repeat

Especially for more advanced folks, perhaps something to which to aspire is being a patch release manager.

So far all of these have been Google employees. But there's been some talk about fixing/improving/enhancing the infrastructure mechanisms that have required that so far. There's a reasonable chance that by the time you're ready for that kind of role that it has opened up.

If nothing else, shadowing the role might be an option in the nearer term. That might also enable you to be the one documenting the role, noting aspects that require Google-internal priv's and not, and perhaps helping find was to spread the patch management load.

Issue Triager

Issue Triager

Image source: https://www.flickr.com/photos/rikkis_refuge/5016931285 (CC BY 2.0)

Every presentation is better with kittens!

Bonus points for these kittens looking like they’re a mix of sleepy, active, focused, zoned out, curious and grumpy...each of those is us software engineers at some point.

Cat Herder

Image source: https://www.flickr.com/photos/rikkis_refuge/5016931285 (CC BY 2.0)

Issue Triager

Does:

- poll bugs, check status
- interact with contributors and SIG leads
  - send reminders
  - ask questions
- publish summary reports

Image source: https://www.flickr.com/photos/rikkis_refuge/5016931285 (CC BY 2.0)

What is it really?

A lot of reading.

Regularly trying to connect with contributors and SIG leads to check for status updates, query outlook on resolution.

Summarizing status up to release team.

A lot of this is work anybody could do, even newcomers.

One particular thing a experienced developer can bring to the release team is intuition for risk: how big is a feature, how late is it, how buggy is it, how much impact does a bug have, is the fix conclusive or a short term fix, etc.? That does involve looking at code, test cases, test results, etc.

Issue Triager

Does not:

- implement fixes/features
- usurp the decision-making power of the SIGs
- look at code...much

Image source: https://www.flickr.com/photos/rikkis_refuge/5016931285 (CC BY 2.0)

What is it really?

A lot of reading.

Regularly trying to connect with contributors and SIG leads to check for status updates, query outlook on resolution.

Summarizing status up to release team.

A lot of this is work anybody could do, even newcomers.

One particular thing a experienced developer can bring to the release team is intuition for risk: how big is a feature, how late is it, how buggy is it, how much impact does a bug have, is the fix conclusive or a short term fix, etc.? That does involve looking at code, test cases, test results, etc.

If not code, then what?

GitHub issues with “v1.XX” milestone label

Slack

Google Groups

Test Grid

SIG Meetings

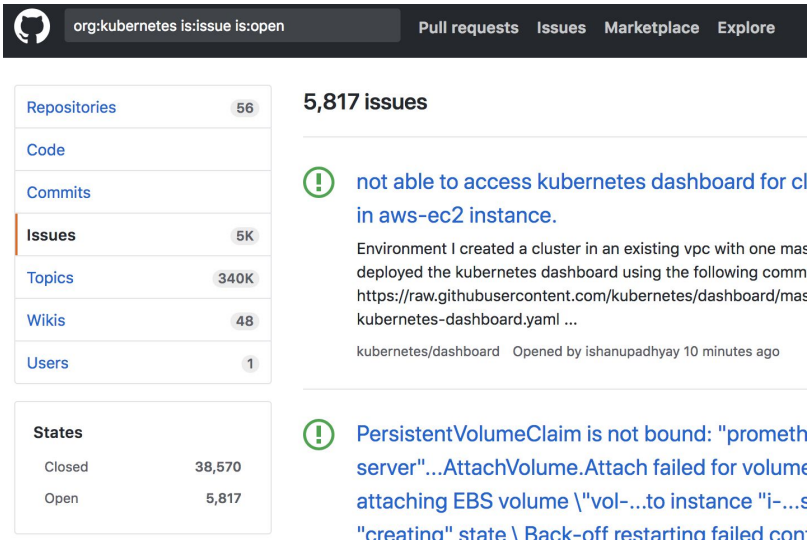
Image source: https://commons.wikimedia.org/wiki/File:Lifeguard_tower_-_San_Agustin_-_Gran_Canaria.jpg (CC BY-SA 3.0)

Reading reading reading

And attending meetings and listening

And some talking

What is an issue?



"Issue" is a GitHub thing.

What is an issue?

Image source: https://www.publicdomainpictures.net/en/view-image.php?image=22032&picture=bucket-and-toys-on-beach (CC0 1.0)

A GitHub issue is just a bucket.

Like most any other source code / project management system.

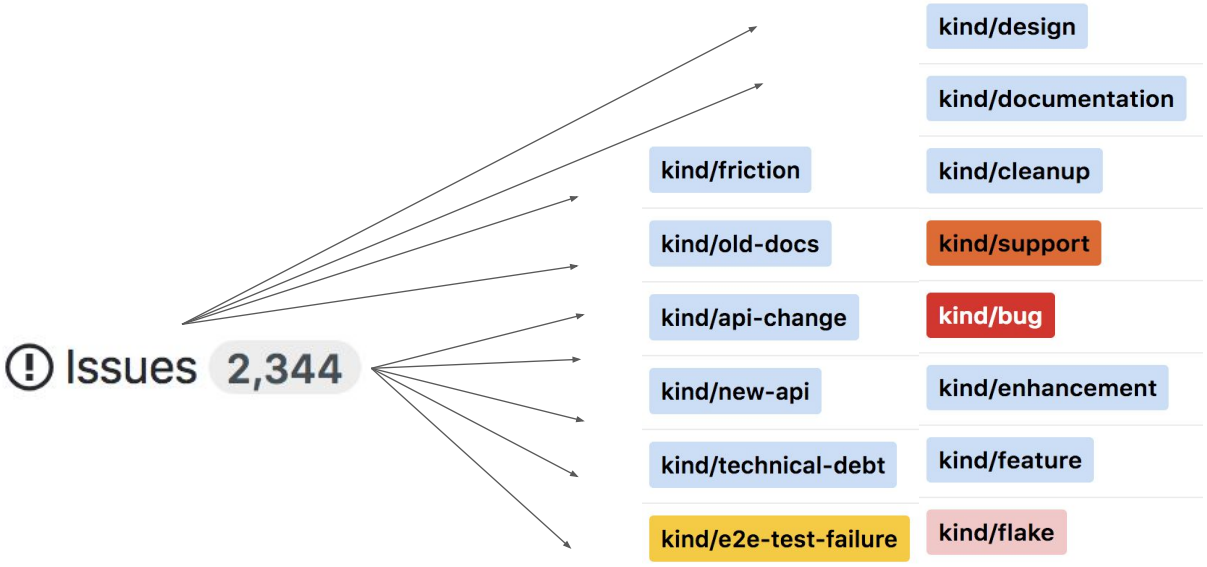
It gets its meaning through the way a project uses it.

What is an issue?

What is an issue?

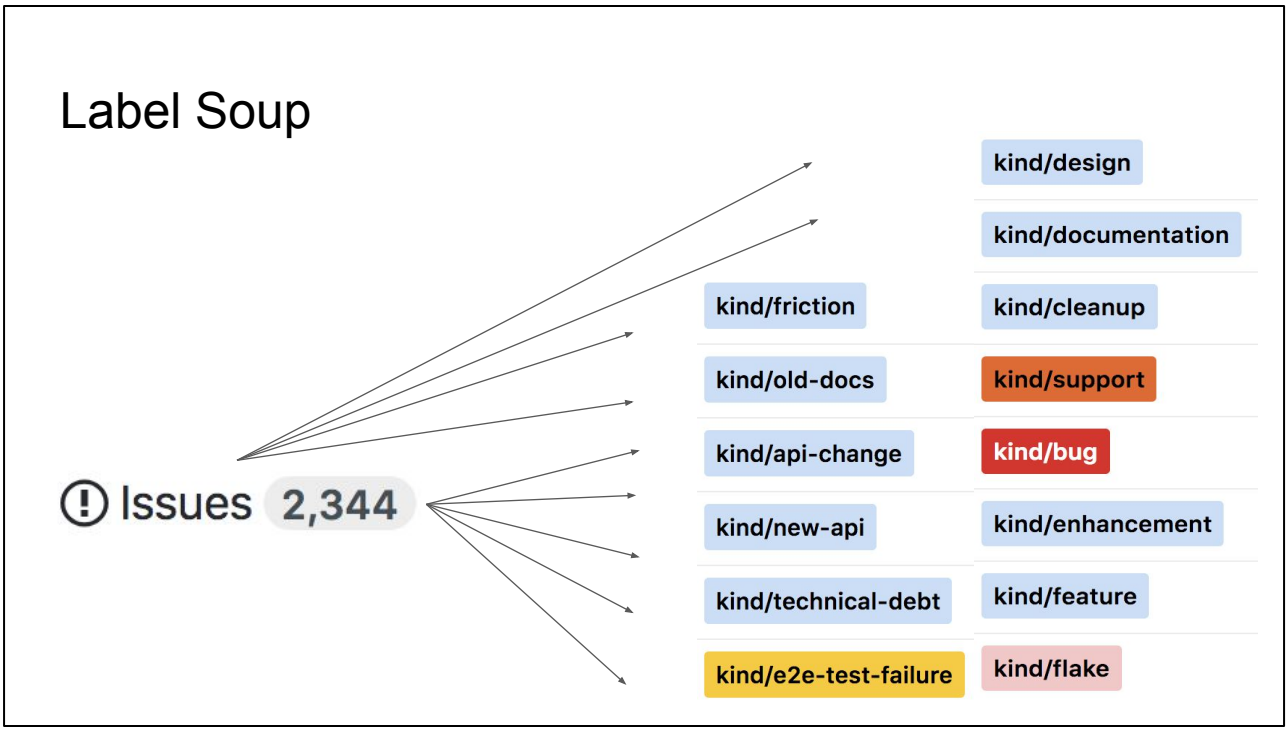
It might be nice if it was two kinds...a bug or a feature.

What is an issue?



But there's more.

So many more.

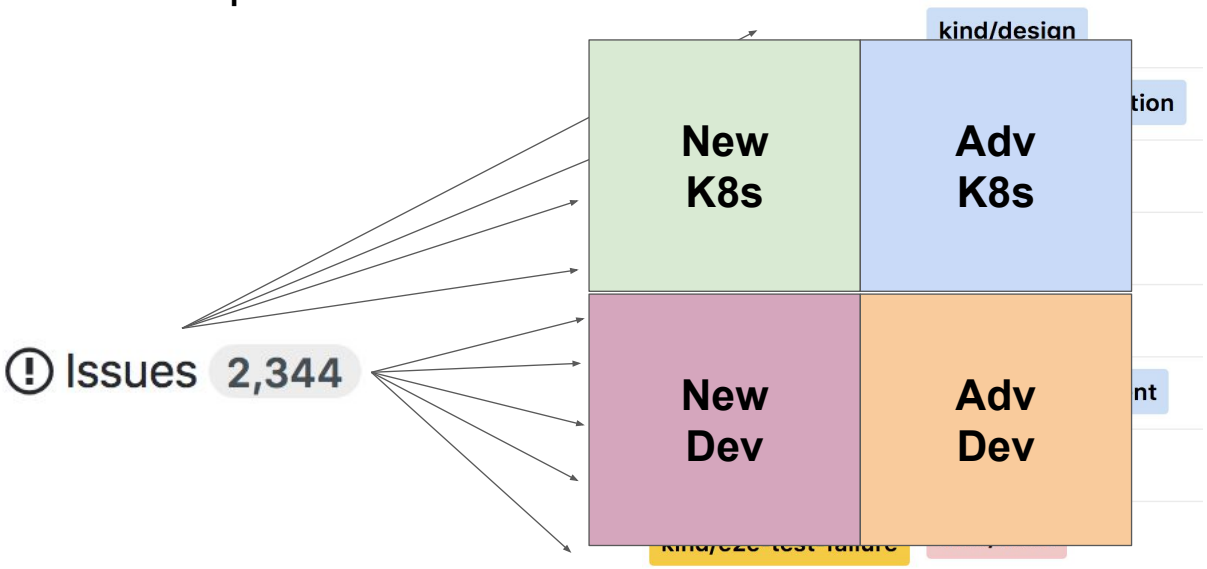


But there's more.

So many more.

Something like 170...down from around 220 or so!

Label Soup



Various tracks of discussion ongoing around making labels consistent across kubernetes repo's, and reducing the number of labels.

"Principle of Least Surprise"

Things need to be simpler, less confusing...no surprises.

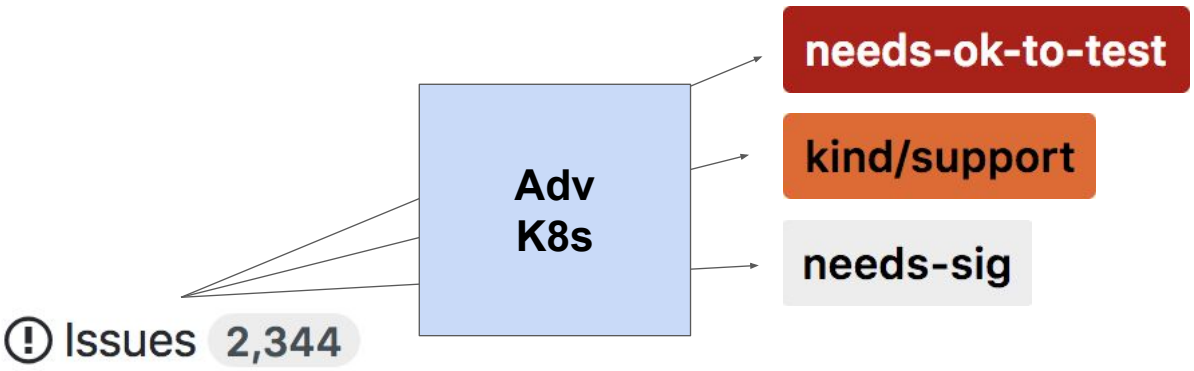
The discussions have been relatively quiet, with a very small number of people involved. Anywhere across the experience spectrum:

If you see something that is confusing or overwhelming, please speak up. More perspectives would be welcome.

If you have an idea for a new label, consider carefully whether it is truly adding some unique value, versus making things more complex and some existing label is sufficient.

If you see an issue that has a wrong or missing label, correct it. If you're not sure, there's community benefit in asking for clarification.

Label Soup



If you see an issue that has a wrong label, correct it. If you're not sure, there's community benefit in asking for clarification.

If you do this for even a few minutes a day or week, you're making a notable impact on triaging issues.

The issue triager(s) for the release team inevitably end up focused primarily on the issues labelled for inclusion in the current milestone (usually a few dozen are open at any time). Help needed to triage the others.

Where do issues come from?

Where do issues come from?

Image source: https://commons.wikimedia.org/wiki/File:Stork_bringing_baby_-_Colmar,_Alsace,_2016.05.18_(35).jpg (CC BY-SA 4.0)

We're in Denmark for KubeCon. How could I not have a reference to Hans Christian Anderson?

Where do issues come from?

Bugs:	humans (inside and outside k8s)	kind/bug
Features:	humans (mostly inside k8s), SIGs	kind/feature
Flake, test-failure:	robots (CI)	kind/flake priority/failing-test

*Issue not required for Pull Request (PR)

...but maybe PR “Fixes #1234”

Issues

During the release cycle, there seems to not be a lot of issues coming from humans. Most of us are risk averse and aren’t running the pre-release builds or head of tree code.

Testing

https://testgrid.k8s.io/

canonical	conformance	google	google-gke-stackdriver
google-kops-gce	istio	kopeio	presubmits
sig-api-machinery	sig-apps	sig-auth	sig-autoscaling
sig-big-data	sig-cli	sig-cluster-lifecycle	sig-gcp
sig-instrumentation	sig-multicluster	sig-network	sig-node
sig-release	sig-scalability	sig-scheduling	sig-storage
sig-testing	sig-ui	tectonic	wg-resource-management

You can click on each of these tiles and get test result details.

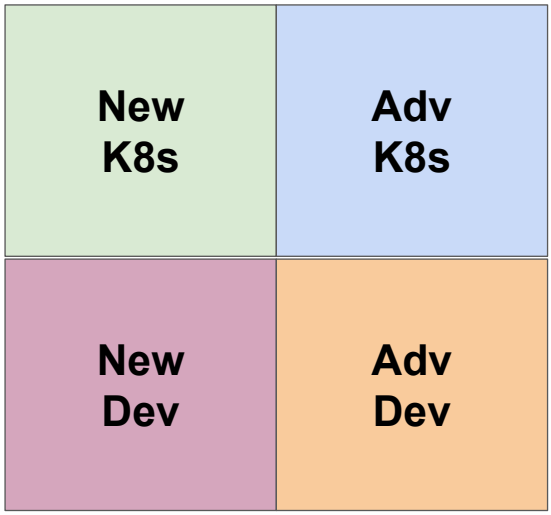
Tracking this is a massive job. Officially it is the job of the CI Signal lead on the release team. But in as much as it is one of the ways issues come to be created, it also overlaps with the issue triager. Need to be able to proactively tunnel in and see trends in test and query results.

Testing

3 main types of test

- Unit
- Integration
- End-to-End (e2e)

Test plan? Everybody's responsibility



At any skill level, analyzing test coverage and adding test cases is valuable.

Both to the project (most SIGs looking for better test coverage), but it also gives you an entry path into code. For an area where you're active already or just curious, golang has great tools for analyzing code coverage of test cases. Areas that are important and lacking test are a great place to contribute.

Understanding tests and the nuances especially of the interactions of integration and e2e tests is a great way to broaden your knowledge of the code base, its design, its weaknesses. Which can also guide you to areas for improvement. In code, in documentation and in k8s project engineering processes.

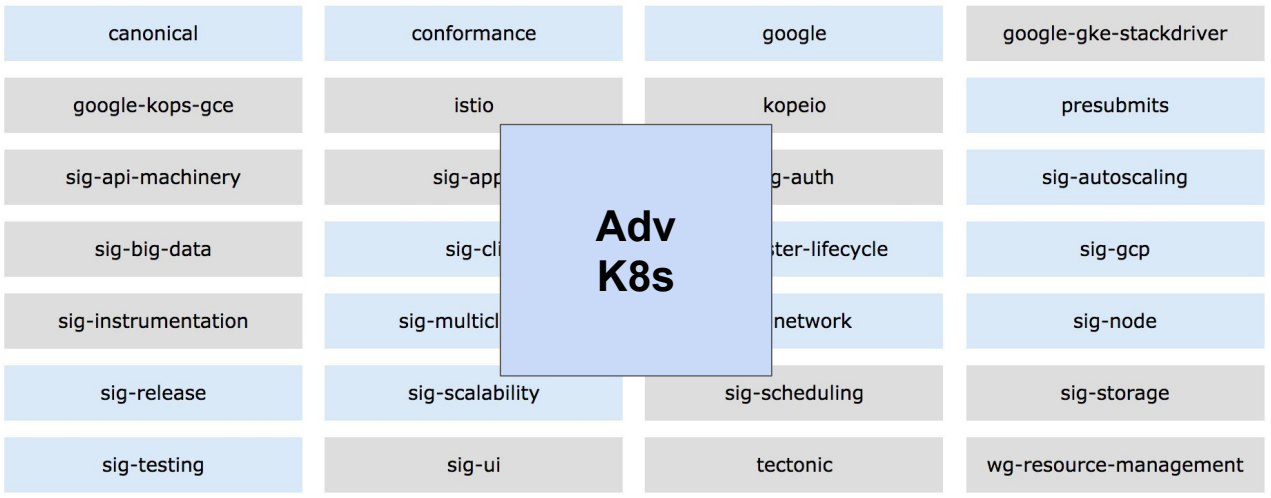
And once you go beyond unit tests, you're also learning how to run some external dependency of k8s for integration testing, or are running a cluster for end-to-end exploration and testing. Running and maintaining a simple cluster is a great way to grow knowledge (although I recognize this can feel daunting at first...k8s is a complex distributed system...see https://github.com/kelseyhightower/kubernetes-the-hard-way for a step by step tutorial on making it happen).

If you've spent time in industry you've probably seen "test plan" documents made by "the QA team". I'm not aware of something like this for k8s and in a distributed open source development way it makes sense. SIG-Testing really owns test infrastructure, but the overall test cases and coverage is owned by "everybody". But in a way there's something like a test plan -lite from the new 1.11 release team CI Signal Lead

(Aish Sundar):
https://docs.google.com/document/d/1uDVrlzh9gZZukz-GYHWSyOnFW_2lplZ4JgR3D4tV4EA/edit#heading=h.x1djb1g2trjc
In documenting how to deal with test issues, Aish paints roughly the picture of what release blocking testing is happening.

Testing

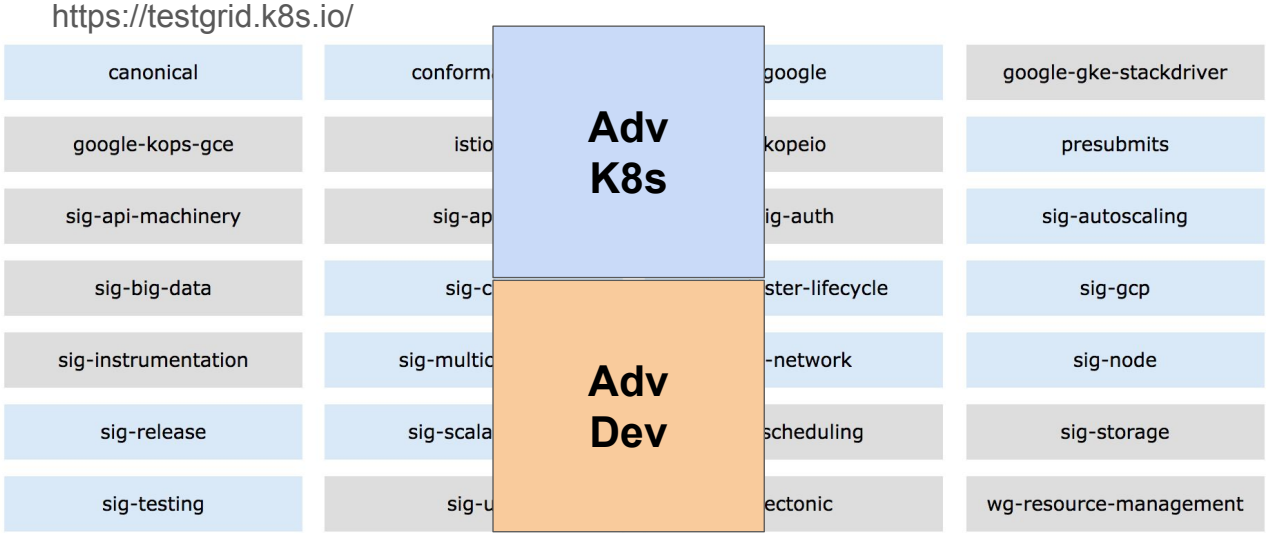
https://testgrid.k8s.io/



This proactive watching and understanding is something anybody can do and which brings value.

Testing

https://testgrid.k8s.io/



@spiffxp has mentioned a given PR (which is known good and should pass testing) has only a 75% chance of testing?!? There are tricks done to make the impact smaller, but still...

There's a lot of cruft in test cases. Complex test cases. Brittle test cases. If you're a skilled engineer and are interested in some chopping wood / carrying water ….this might be an impactful place!

SIGs SIGs SIGs (and WGs too)

As issue triager need to have a broad overview of:

- which SIGs do what
- who's who
- risk level for SIG's feature set
- current health of SIG code

Architecture	architecture	* Brian Grant, Google * Jaice Singer DuMars, Microsoft	* Slack * Mailing List	* Regular SIG Meeting: Thursdays at 15:30 UTC (weekly)
Auth	auth	* Eric Chiang, Red Hat * Jordan Liggitt, Red Hat * Tim Allclair, Google	* Slack * Mailing List	* Regular SIG Meeting: Wednesdays at 11:00 PT (Pacific Time) (biweekly)
Autoscaling	autoscaling	* Marcin Wielgus, Google * Solly Ross, Red Hat	* Slack * Mailing List	* Regular SIG Meeting: Mondays at 14:00 UTC (biweekly/triweekly)
AWS	aws	* Justin Santa Barbara * Kris Nova, Heptio * Bob Wise, AWS	* Slack * Mailing List	* Regular SIG Meeting: Fridays at 9:00 PT (Pacific Time) (biweekly)
Azure	azure	* Jason Hansen, Microsoft * Cole Mickens, Red Hat * Jaice Singer DuMars, Microsoft	* Slack * Mailing List	* Regular SIG Meeting: Wednesdays at 16:00 UTC (weekly)

https://git.k8s.io/community/sig-list.md

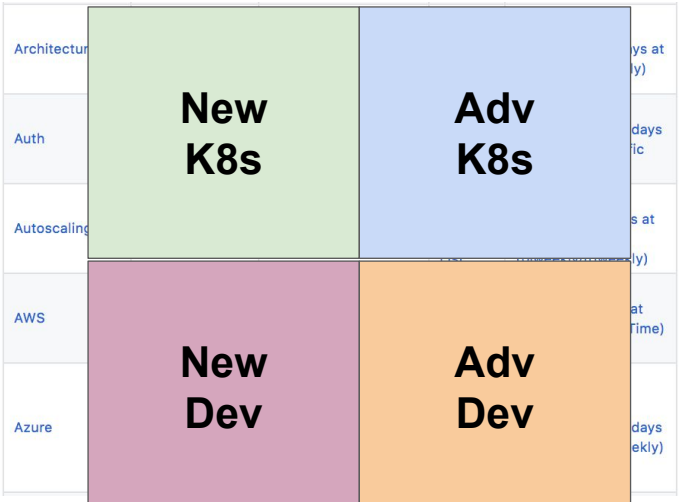
Test grid output almost looks like a SIG list, which points out...

Issue triage takes broad understanding of SIGs

SIGs SIGs SIGs (and WGs too)

As issue triager need to have a broad overview of:

- which SIGs do what
- who's who
- risk level for SIG's feature set
- current health of SIG code



https://git.k8s.io/community/sig-list.md

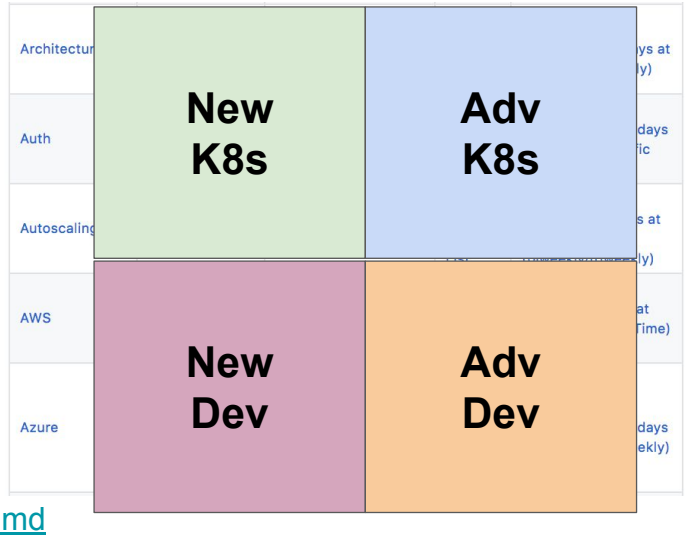
Again it's a great way to broaden your knowledge.

Depth of understanding of each SIG/WG isn't required. But broadening your knowledge of SIG/WG activities can be your entry point to becoming more involved.

So how do you learn about a SIG...

SIGs SIGs SIGs (and WGs too)

***** Charter Documents**



https://git.k8s.io/community/sig-list.md

SIG charters! Follow their establishment...roles and responsibilities shall be documented.

“Can I take on leading that role?”

“Can I shadow that role?”

Meetings Meetings Meetings

Meetings Meetings Meetings

Image source:
https://www.google.com/search?&tbm=isch&q=are+you+lonely+hold+a+meeting+the+practical+alternative+to+work&oq=are+you+lonely+hold+a+meeting+the+practical+alternative+to+work

...unknown original author

https://8wdee.files.wordpress.com/2013/08/hold-a-meeting.jpeg

Meetings have a bad reputation...don't hate on meetings….

Meetings Meetings Meetings

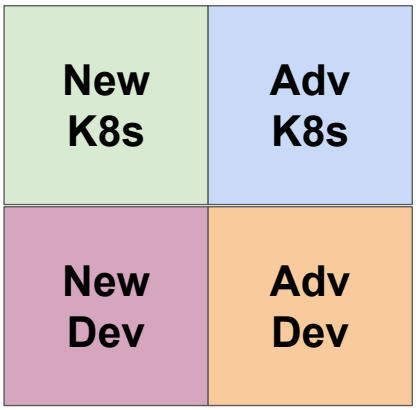
THE calendar: https://kubernetes.io/community/

Initial (and ongoing) overview and orientation:

- ***THE*** Community Meeting
- http://bit.ly/k8scommunity
- Thursdays on zoom.us

Release, Steering, Architecture

SIG/WG: Each has meeting(s), open to all



Kubernetes is made by meetings!

Yes the volume of different meetings can be daunting.

The main calendar shows them all. That calendar links to the info to join meetings and also their agendas and meeting minutes. Reading through those is a great way to learn more about a SIG in which you have an interest to know more.

The community meeting is another way to learn about SIGs. Each week a few are highlighted, with a representative attending the community meeting to brief the attendees on recent and upcoming activities.

Release team meetings are weekly, and then multiple times a week, and then daily, as the release process progresses. Attending these can give you insights into the overall process.

Also the Steering committee meetings are recorded and published to the k8s youtube channel:
Search “kubernetes steering committee meeting”. Also
https://github.com/kubernetes/steering

And SIG-Architecture: The Architecture SIG maintains and evolves the design principles of Kubernetes, and provides a consistent body of expertise

necessary to ensure architectural consistency over time.

All this gives you additional overarching information about the broader project, and is the level of knowledge you get working on the release team.

Of course each SIG/WG has meetings too. If you've got, or find while working through your learning, a focused interest in a particular SIG...by all means start attending its meetings regularly. If it's a bad time, there are recordings. If it's a bad time, suggest to the SIG that they vary meeting times to be more inclusive. Don't be shy. SIGs welcome agenda items and many SIGs have fairly sparse agendas. So there's probably time to ask even a newbie question. Don't be shy. It's good for SIGs to know they have newbies and where they struggle. Finding your voice and being present and engaged in SIG meetings can be a path to finding more areas of contribution in a SIG.

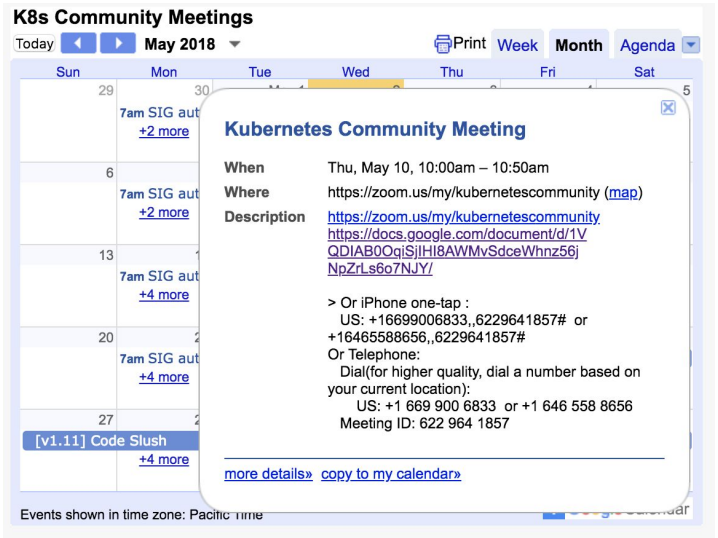
As a newbie on the release team I had to occasionally attend SIG meetings to get information. It felt a bit intimidating at first, but that was all in my mind. Each SIG has been inclusive and inviting and positive.

Every SIG meeting is a chance for learning. Yesterday in contrib-summit some sig's mentioned they're are increasingly using some of their SIG meetings as learning/teaching/discussion time to talk about code, designs, architecture, and implementation. Attend these meetings and soak it up. Find your voice...Just like in my release team role where I had to show up unannounced and unknown and ask the occasional question...ask questions. Use your "newbie" position to ask a question. Especially if the SIG isn't doing this...ASK FOR IT! …"hey could we have a session covering the overall design/architecture of FOO...and I can note take during the session and bulk up our code documentation...win win?"

I really believe it's on individual leaders and leaders in individual SIGs and WGs to facilitate this type of learning. Which leads to...

Meetings Meetings Meetings

Meetings Meetings Meetings



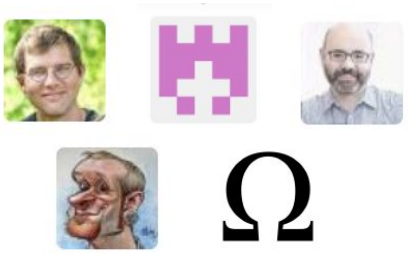
Mentor Program

Communities grow as time goes by...



Mentor Program

Communities grow as time goes by...

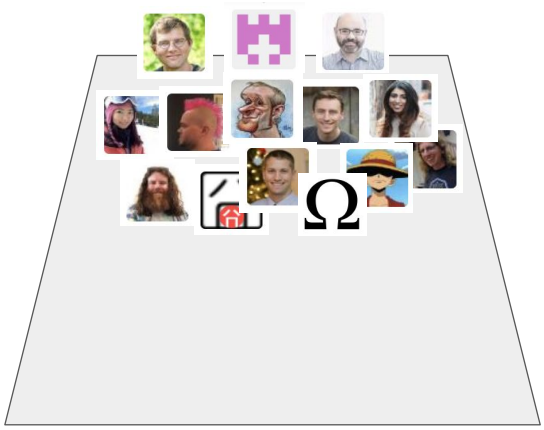


Mentor Program

Communities grow as time goes by...

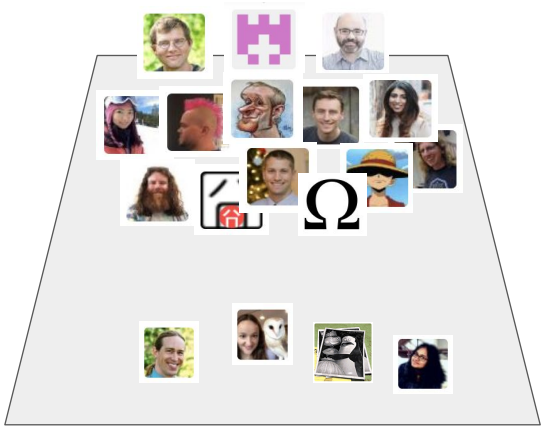
Mentor Program

Communities grow as time goes by...



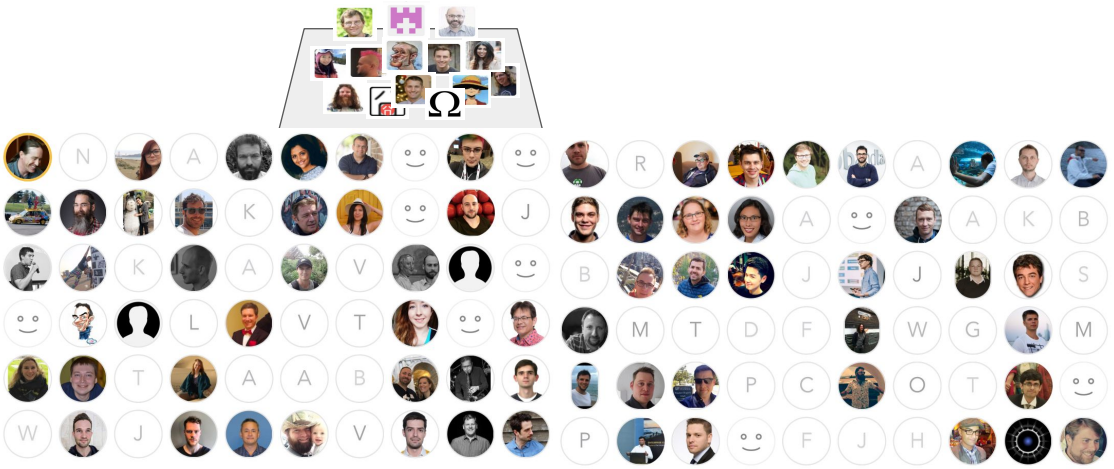
Mentor Program

Communities grow as time goes by...



Mentor Program

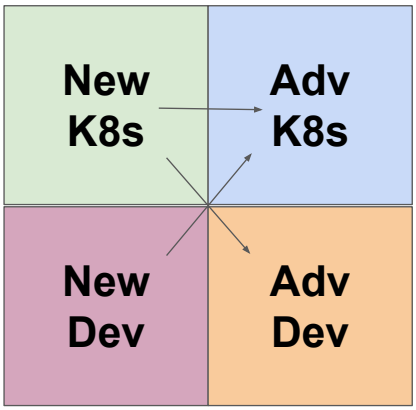
Communities grow as time goes by...



Mentor Program

Communities grow as time goes by…

When more senior members mentoring newcomers



Spread load

Sustainable community

Avoid burnout

Succession planning!

Mentor Program

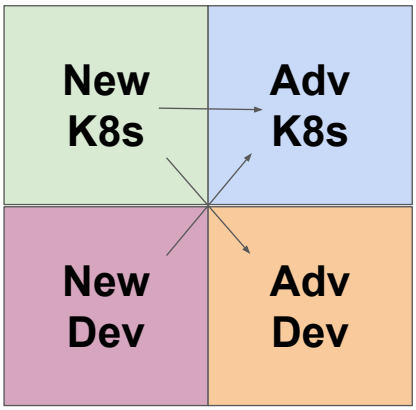
[Meet Our Contributors]

[Group Mentoring Cohorts]

[The 1:1 Hour]

[Google Summer of Code]

[Outreachy]



Most every sig / wg could use addition folks actively helping and contributing.

More contributors are needed across experience levels for project sustainability and avoiding burnout.

1.10 release cycle saw a few sigs that were struggling for active membership and also a few SIGs turning over leaders. New members spread the load.

(new) Contributors->Reviewers

Reviewers->Approvers

Approvers->Owners and SIG leads

It feels like the kubernetes project is normal point in a maturity cycle where some long time contributors and leads might reasonably take a break or move on to something else. You can see this if you look at the developer statistics on github. There are people who were heavy contributors in 2015 and 2016 and less so. There are people who weren't present until recently and are becoming more and more active. There is a natural cycle here.

This presents opportunity for new folks who are establishing themselves in the community to move up to increased responsibility.

Go grow and do go k8s things!

Pick a focus SIG (or two?)

Attend their meetings

Follow their issues and PRs

Learn their code, test cases, designs, weaknesses, strengths

Contribute to the issue backlog, “help wanted”, “good first issue”

Find your voice, ask questions, write down the answers...then teach others

Ask for mentoring on issues that are just beyond your ability...then teach others

Thank You!

SIG contributor-experience
SIG release
Davanum Srinivas
Carolyn Van Slyck
Guinevere Saenger
Paris Pittman
Leah Petersen
Tim Hockin
Jaice Singer DuMars
Aaron Crickenberger
Teague Cole
Jorge Castro
Josh Berkus
Noah Abrahams

vmware

People and SIGs who don’t realize it but contributed to this talk.

Oh and my employer who pays me to do cool open source stuff.