
KubeCon
CloudNativeCon
North America 2023


KubeCon | CloudNativeCon

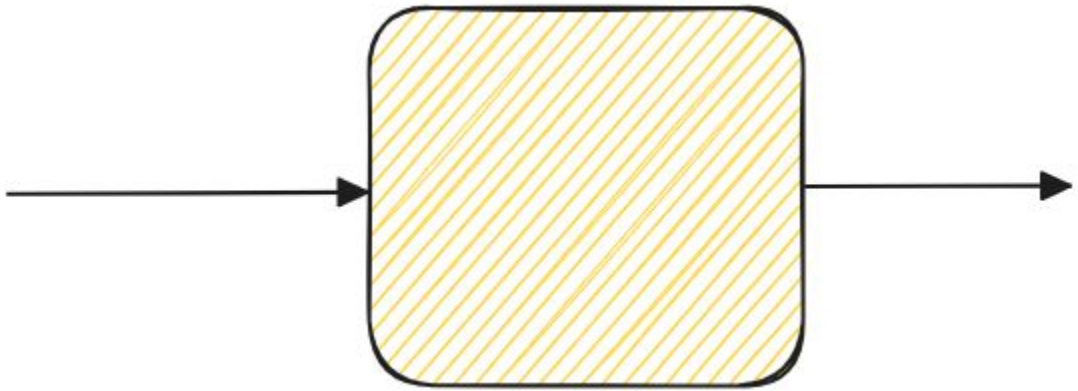
North America 2023

The Eight Fallacies of Distributed Cloud Native Communities

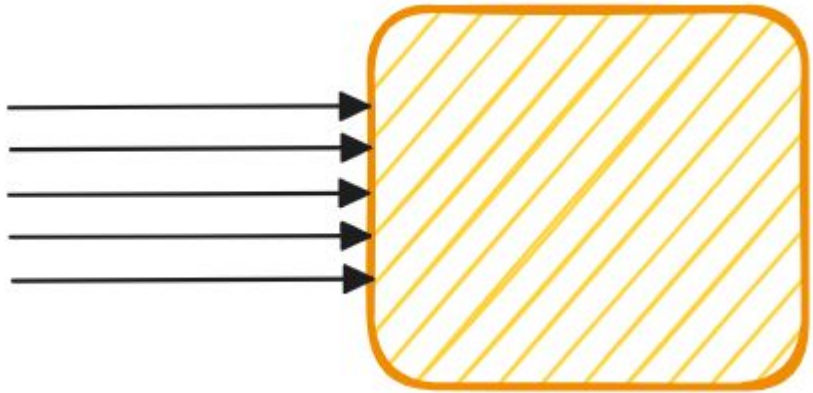
Madhav Jivrajani, VMware
Nabarun Pal, VMware

Distributed Systems

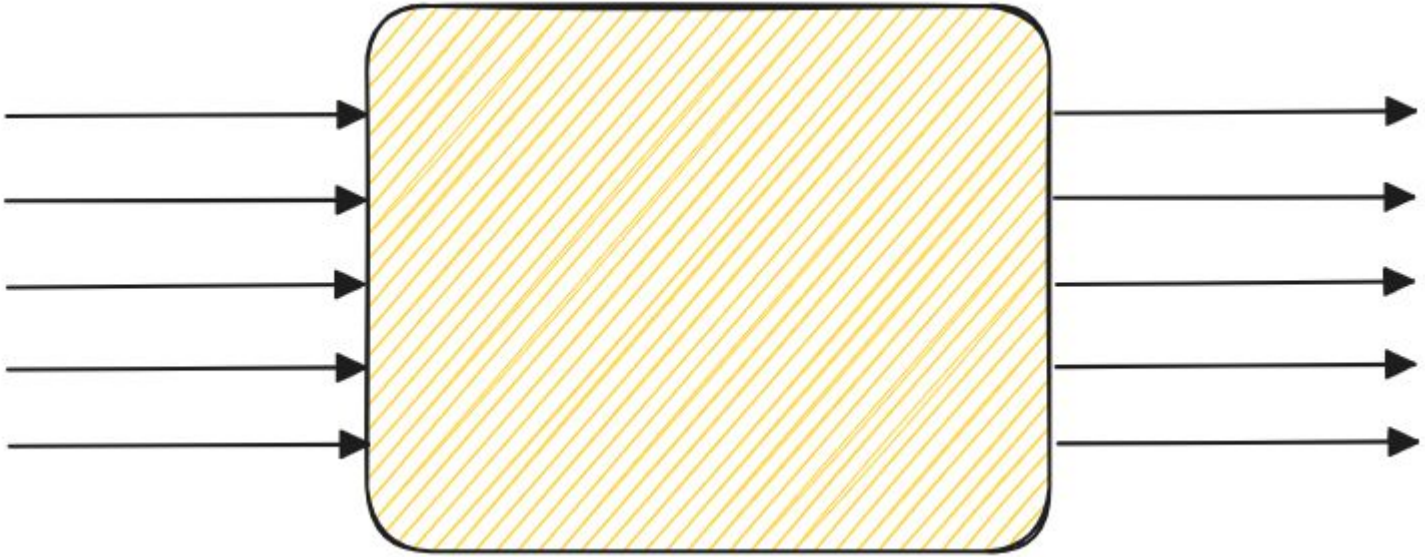
Distributed Systems



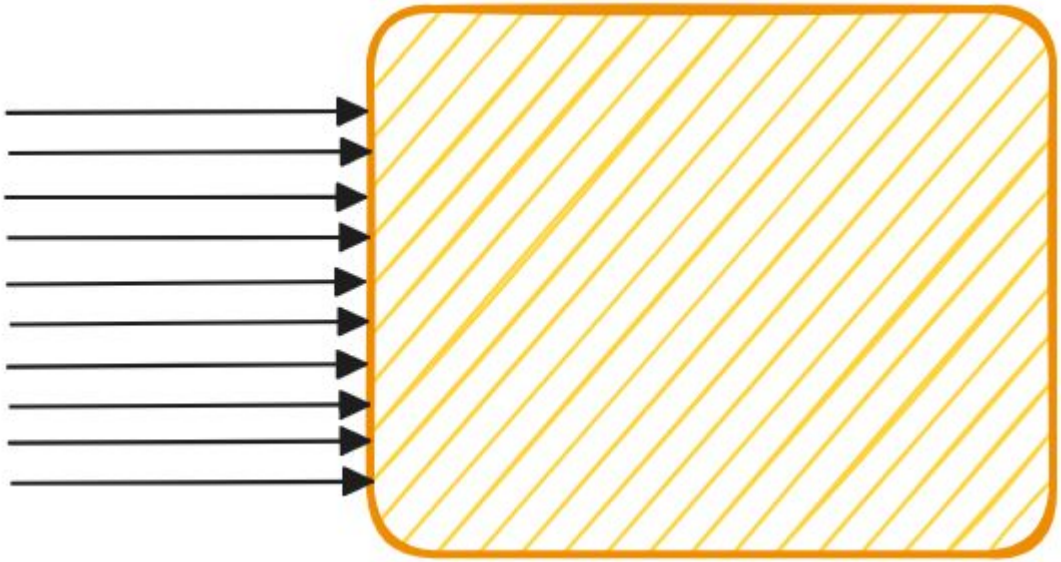
Distributed Systems



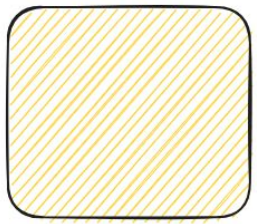
Distributed Systems

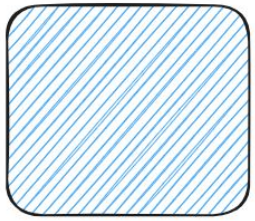


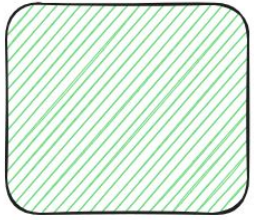
Distributed Systems

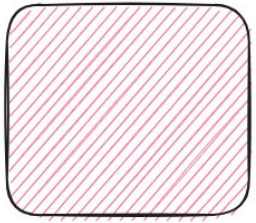


Distributed Systems

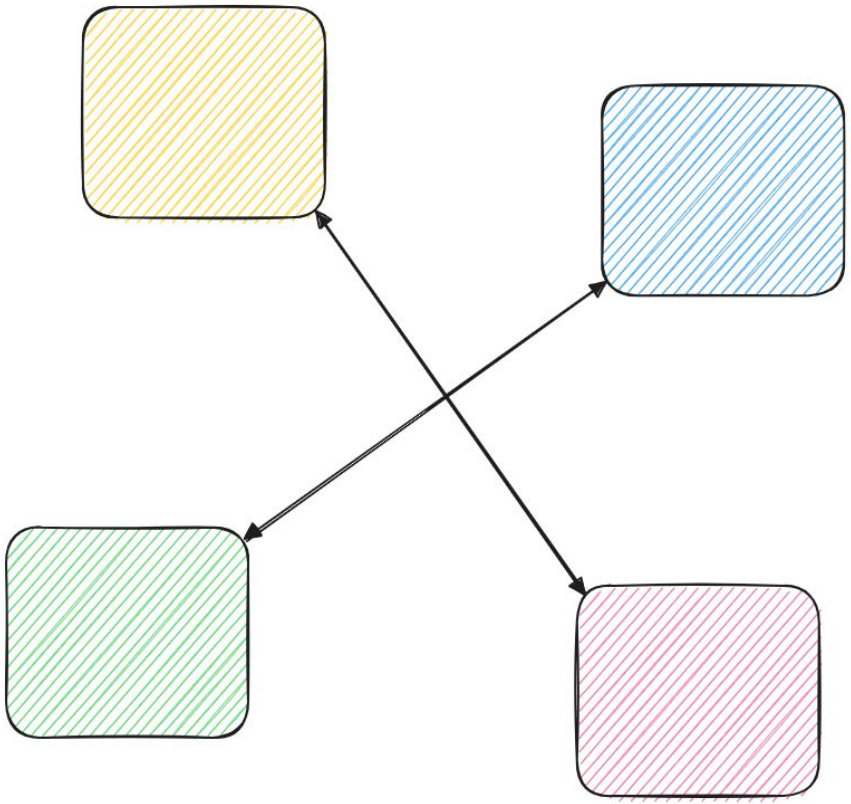




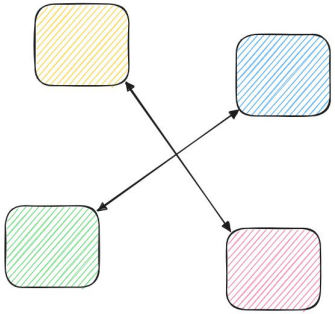


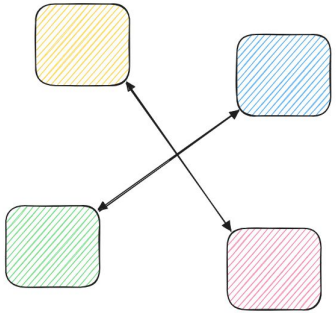


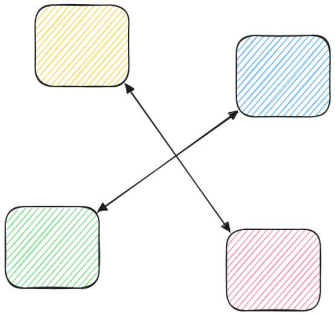
Distributed Systems

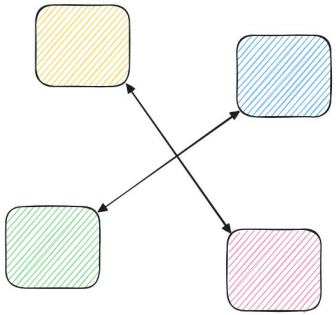


Distributed Systems

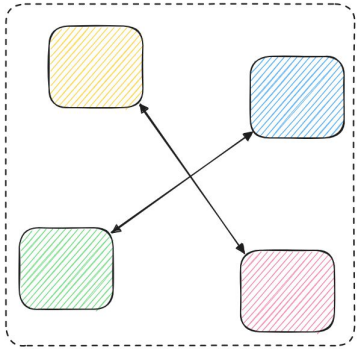


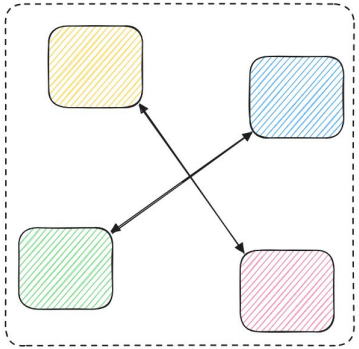


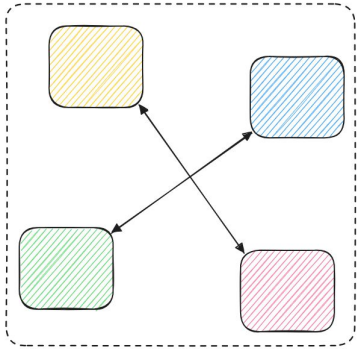


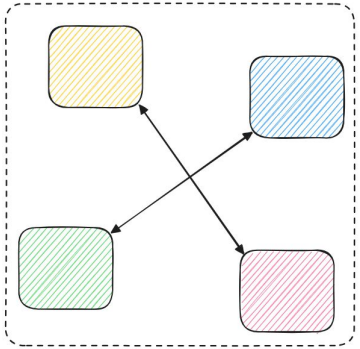


Distributed Systems

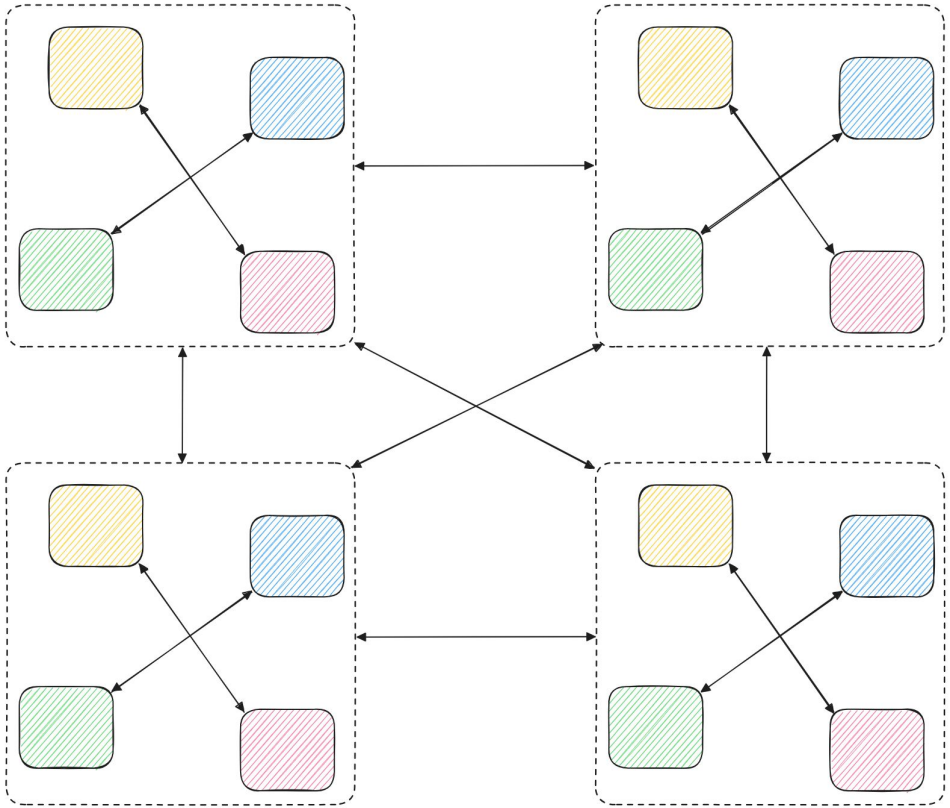








Distributed Systems

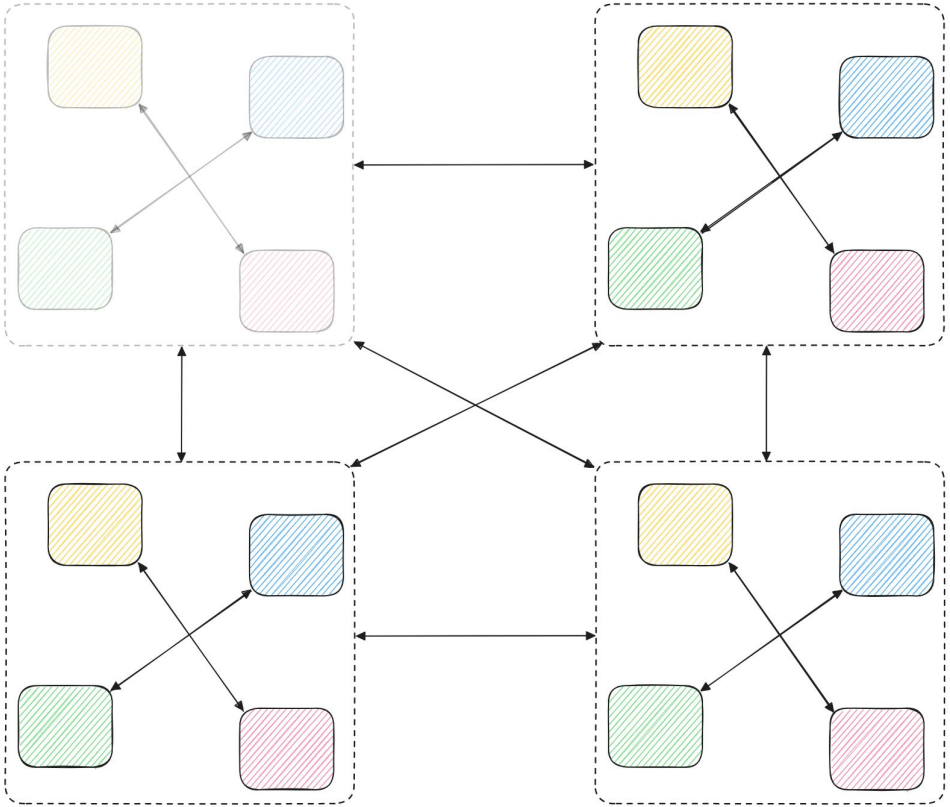


Distributed Systems

Having a globally distributed set of machines, talking over a network gives us all kinds of nice benefits!

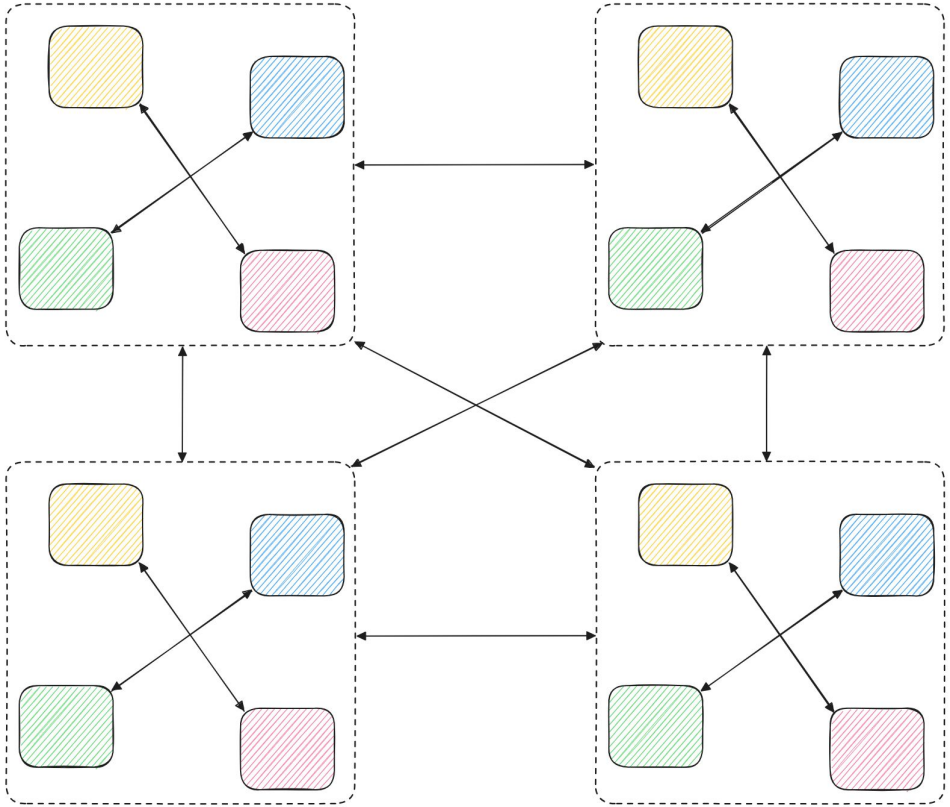
Distributed Systems

If one set of machines are unavailable, we still continue working and making progress towards a shared goal.



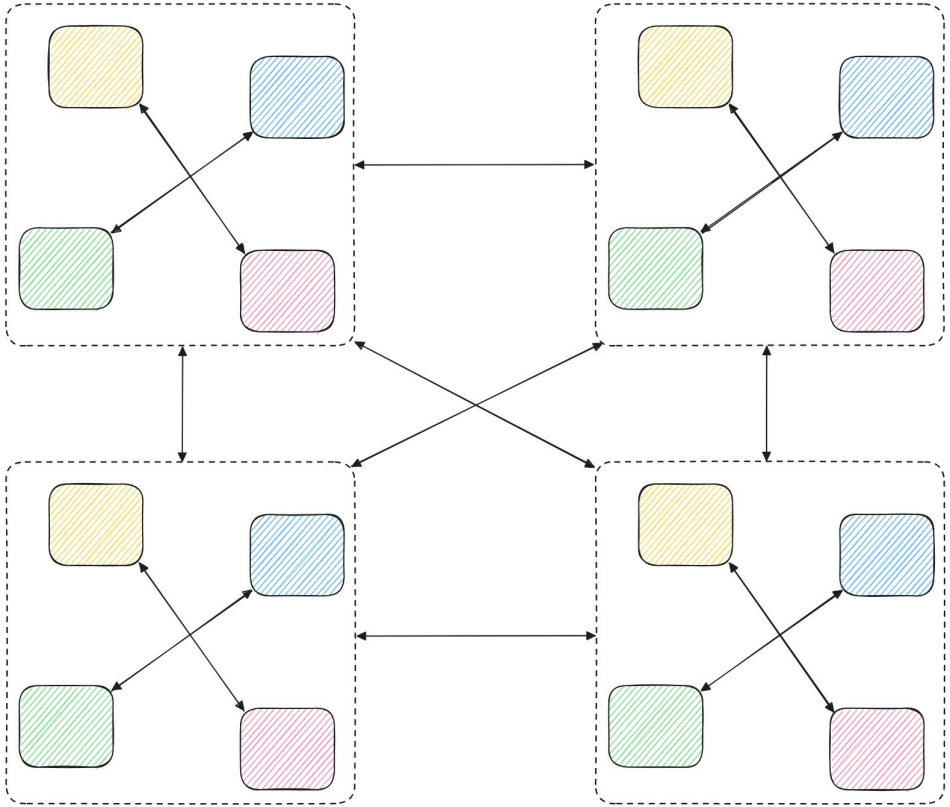
Distributed Systems

Not all machines need to be specialised to do the same thing, each can be meant for a subset of tasks needed to achieve the shared goal.



Distributed Systems

Machines can work parallelly and get more done in the same amount of time without needing to have synchronous communication.



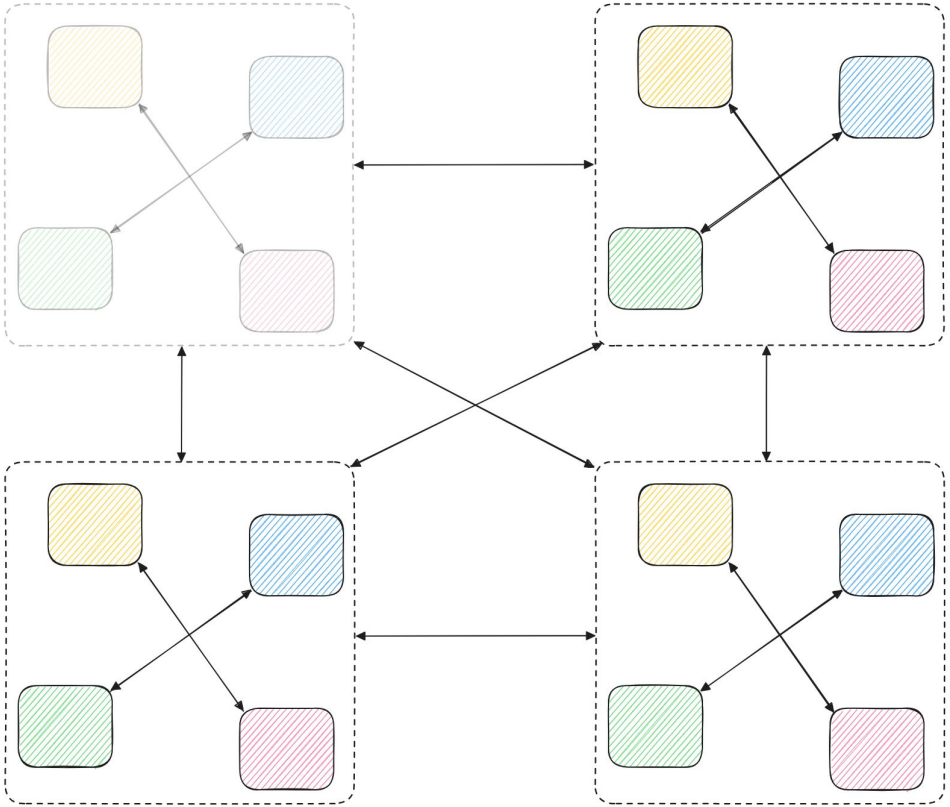
Distributed Systems

But there's no free lunch. With all the niceness, there also comes a slew of challenges!

Distributed Systems

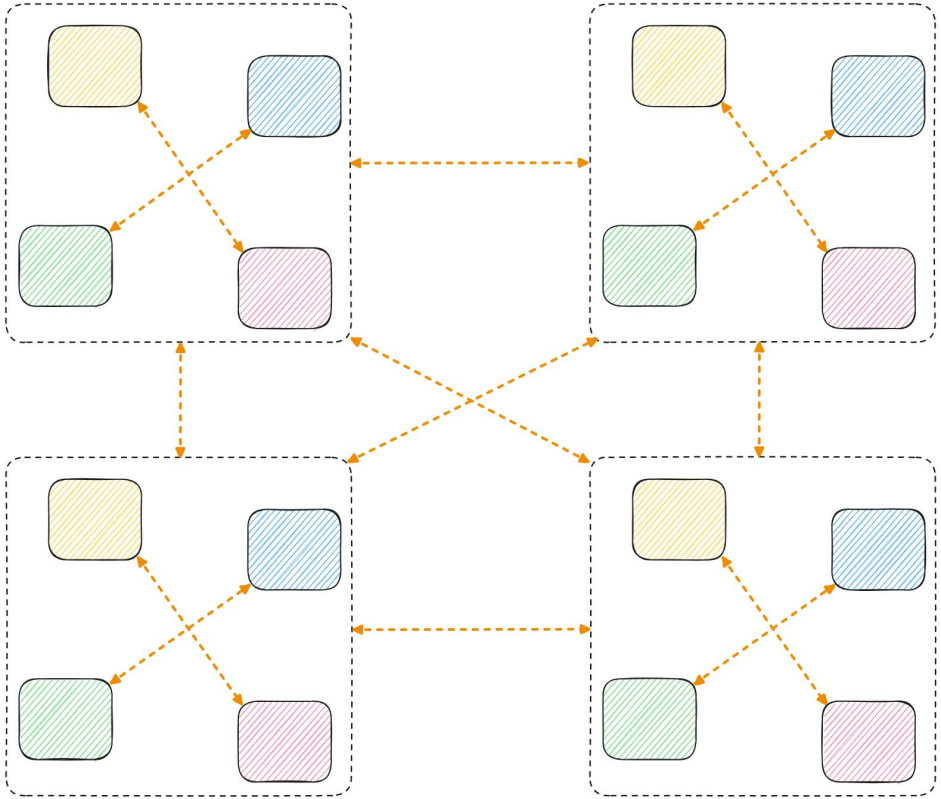
When things go wrong, who fixes them?

How does the system heal?



Distributed Systems

Communications arrive super late, and sometimes not at all, and due to no fault of anyone or anything.



Distributed Systems

As our system grows, so does its complexity and the challenges that come with it.

Distributed Systems

These challenges all exist because we work with a globally distributed set of heterogeneous machines.

Distributed Systems

But it is exactly this set of challenges and the niceties we know we can have that make Distributed Systems a really elegant and beautiful field of study.

Distributed Systems

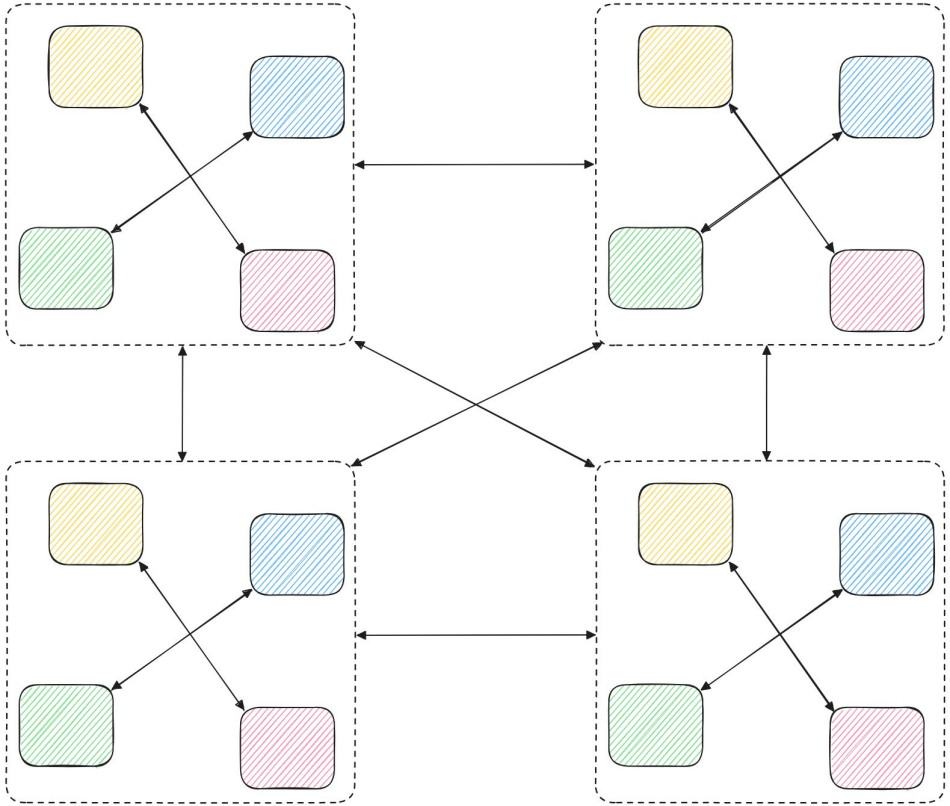
Interestingly enough, most of these challenges are not solvable. In fact the “formal” name for some of them are “impossibility results”.

Distributed Systems

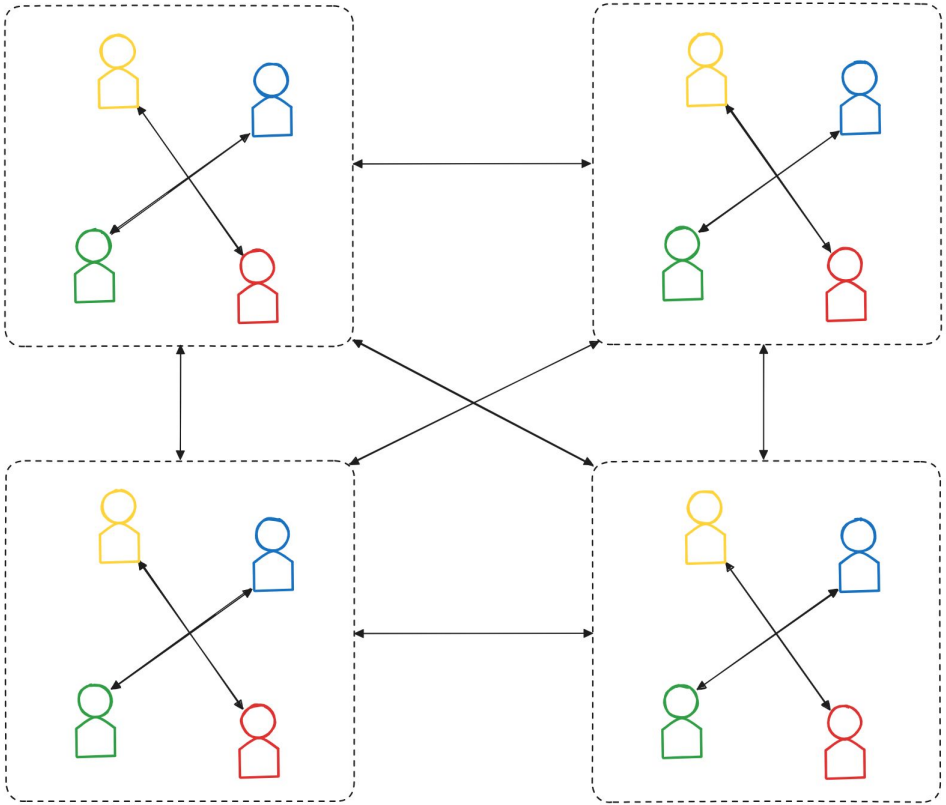
What is important however, and often the solution, is understanding and acknowledging that these challenges exist.

Cloud Native Communities

Cloud Native Communities

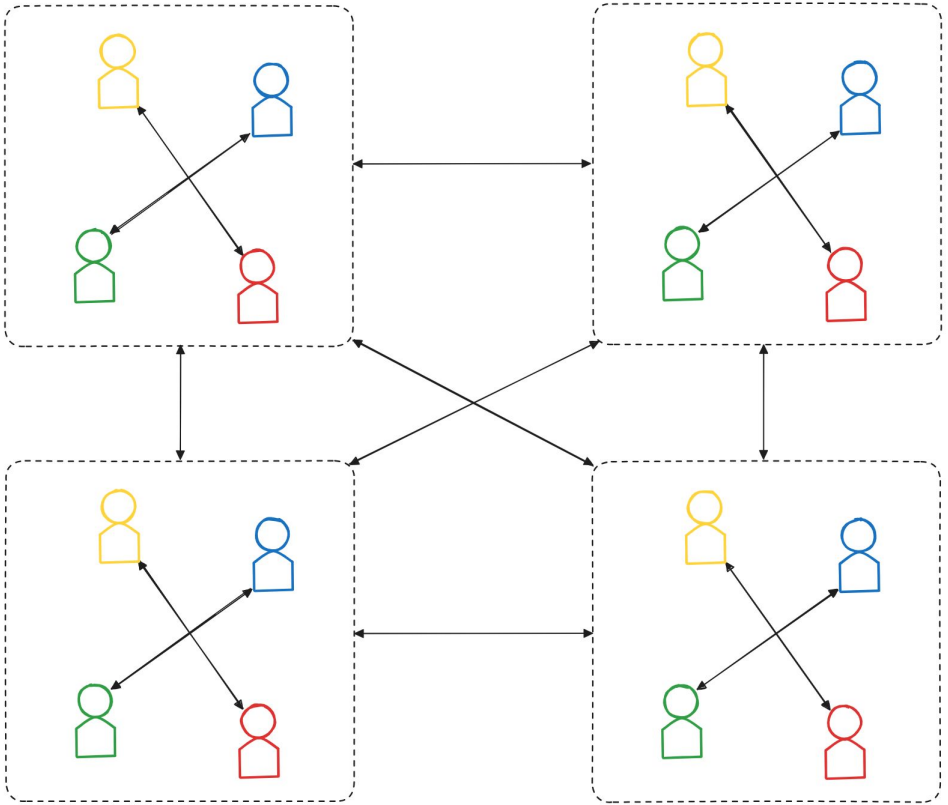


Cloud Native Communities



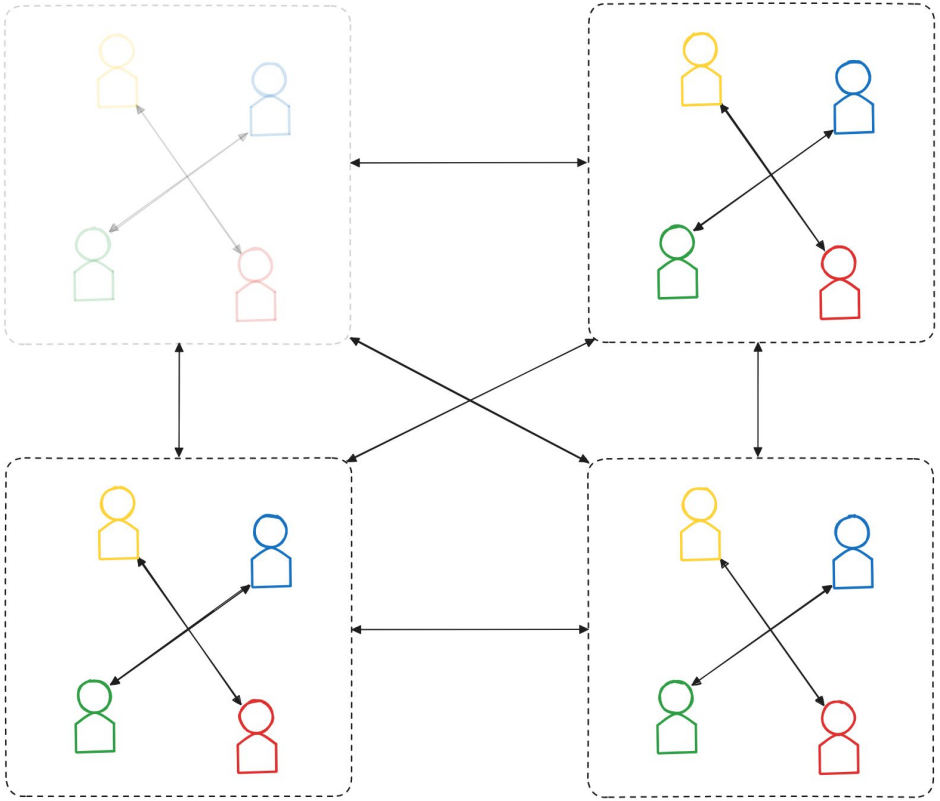
Cloud Native Communities

- Here you have a set of globally distributed people, all collaborating towards a common goal!



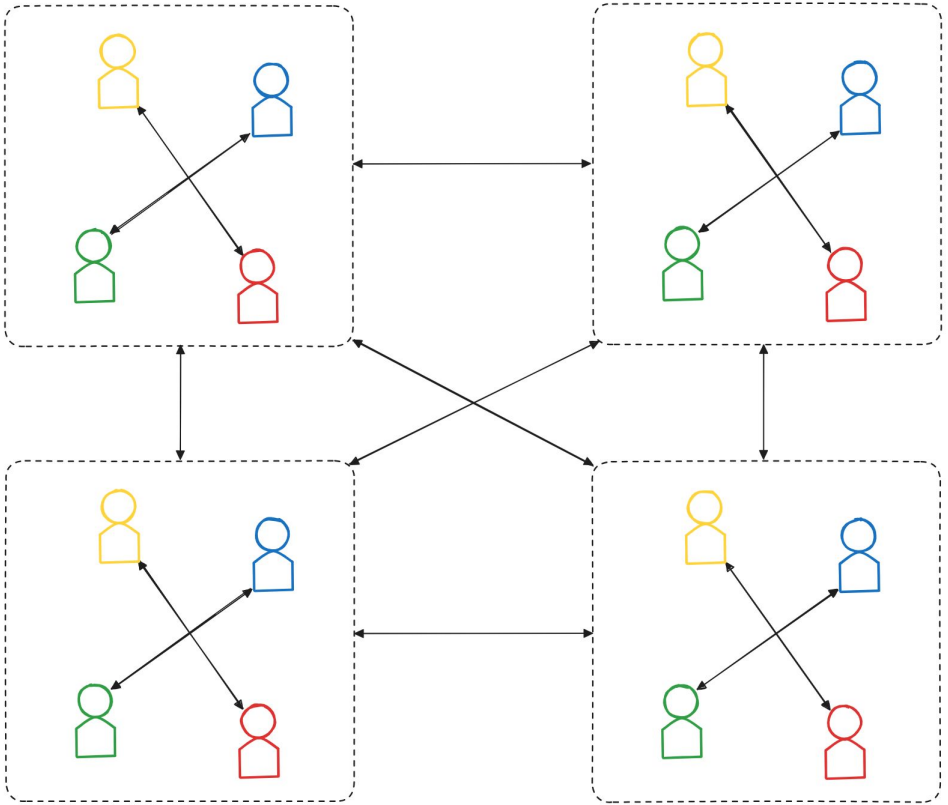
Cloud Native Communities

- Here you have a set of globally distributed people, all collaborating towards a common goal!
- Again, some folks can become unavailable, but that's alright! We help each other out.



Cloud Native Communities

- Here you have a set of globally distributed people, all collaborating towards a common goal!
- Again, some folks can become unavailable, but that's alright! We help each other out.
- Here too, folks can continue working in parallel.



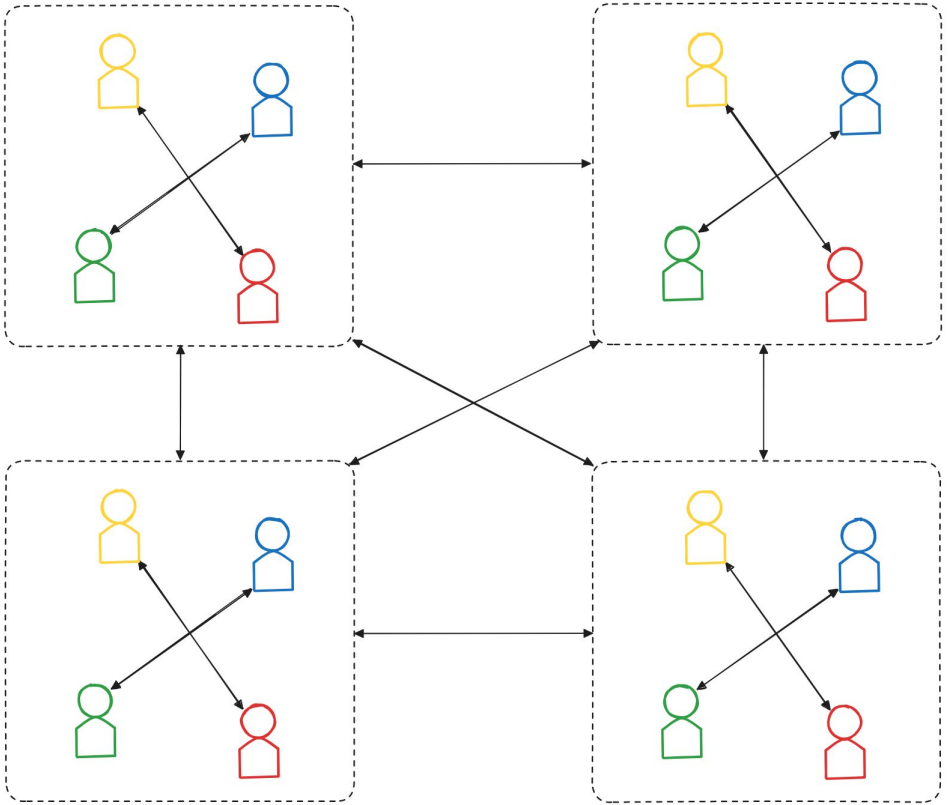
Cloud Native Communities

Again, with all the niceties, we also get a bunch of challenges! Challenges that are arguably more difficult to solve.

Cloud Native Communities

- Maintainer burnout.
- Onboarding new contributors.
- Time zone differences and language barriers.

… and many more.



Cloud Native Communities

As before, some or even most of these challenges are not solvable.

Cloud Native Communities

But our jobs are maintainers, contributors or end-users is to understand and acknowledge these challenges while exercising empathy and kindness.

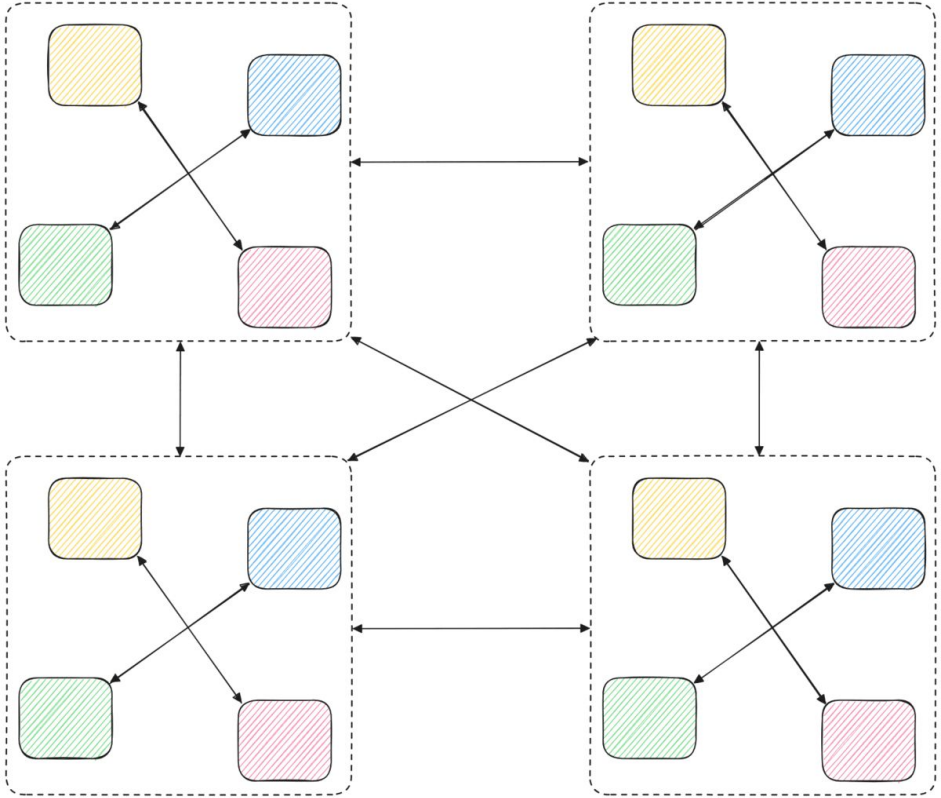
Cloud Native Communities

As our community grows, so does its complexity and the challenges that come with it.

Distributed Systems + Cloud Native Communities?

Distributed Systems + Cloud Native Communities?

Needless to say, there are similarities between the two.



Navigating Complexity By Knowing What Not To Do

Navigating Complexity By Knowing What Not To Do

As distributed systems started becoming mainstream and their complexity grew, a set of *fallacies* were introduced to act as guidelines for common pitfalls one might face.

Navigating Complexity By Knowing What Not To Do

*The **fallacies of distributed computing** are **a set of assertions** made by L Peter Deutsch and others at Sun Microsystems **describing false assumptions** that programmers **new to distributed applications invariably make**.*

Navigating Complexity By Knowing What Not To Do

The Eight Fallacies of Distributed Systems

The **network** is reliable

Latency is zero

Bandwidth is infinite

The network is **secure**

Topology doesn't change

There is one **administrator**

Transport cost is zero

The network is **homogeneous**

Navigating Complexity By Knowing What Not To Do

Similar to this, as our Cloud Native Communities grow, evolve, and become rightfully more complex, we need a set of fallacies to help us navigate it and better sustain and support it.

Navigating Complexity By Knowing What Not To Do

The Eight Fallacies of Distributed Cloud Native Communities

Navigating Complexity By Knowing What Not To Do

The Eight Fallacies of Distributed Cloud Native Communities

The **network** is reliable

Latency is zero

Bandwidth is infinite

The network is **secure**

Topology doesn't change

There is one **administrator**

Transport cost is zero

The network is **homogeneous**

Navigating Complexity By Knowing What Not To Do

The Eight Fallacies of Distributed Cloud Native Communities

The **network** is reliable

Latency is zero

Bandwidth is infinite

The network is **secure**

Topology doesn't change

There is one **administrator**

Transport cost is zero

The network is **homogeneous**

Timelines are reliable

Feedback loops are tight

Maintainer **bandwidth** is infinite

Software supply chain is **secure**

Commitments don’t change

Compromise is a rarity and not the norm

Cost of **sustainably onboarding contributors** is zero

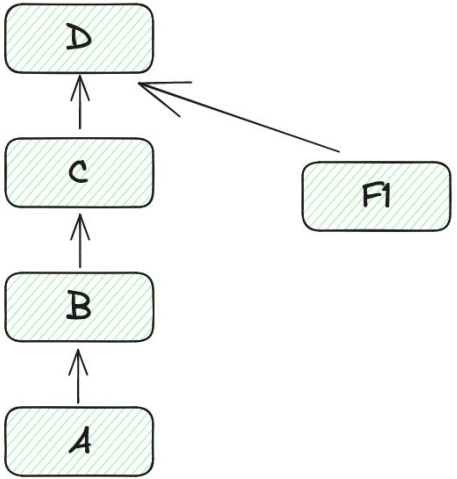
Staffing across project areas is **homogenous**

Fallacy #1: Timelines Are Reliable

The network is reliable: Software applications are written with little error-handling on networking errors. *During a network outage, such applications may stall or infinitely wait for an answer packet, permanently consuming memory or other resources. When the failed network becomes available, those applications may also fail to retry any stalled operations or require a (manual) restart.*

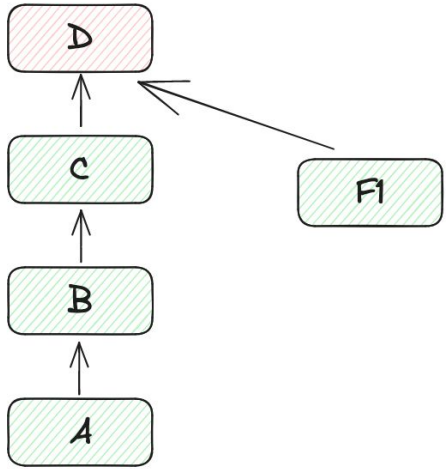
Fallacy #1: Timelines Are Reliable

People expect that the quality of every merge to the code will be same.



Fallacy #1: Timelines Are Reliable

However, that's not the case.

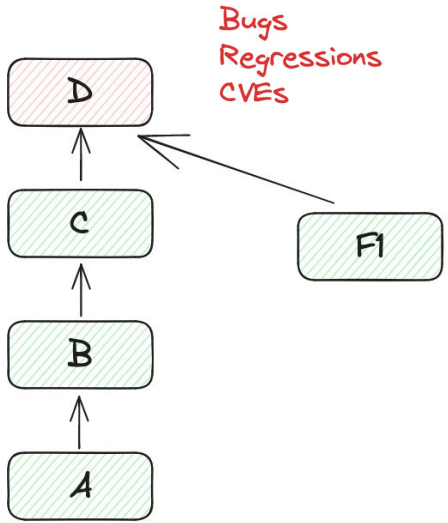


Fallacy #1: Timelines Are Reliable

“Anything that can go wrong will go wrong.”

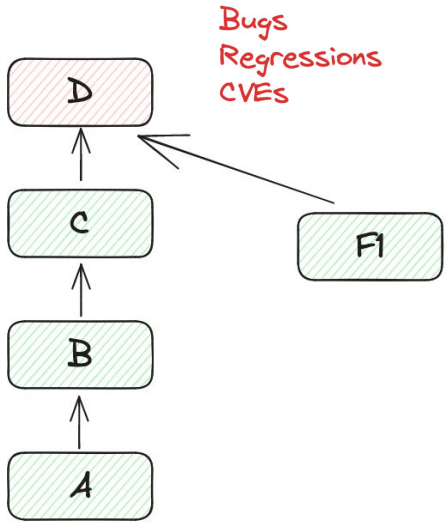
Fallacy #1: Timelines Are Reliable

There can be bugs, regressions and vulnerabilities associated with the new code.

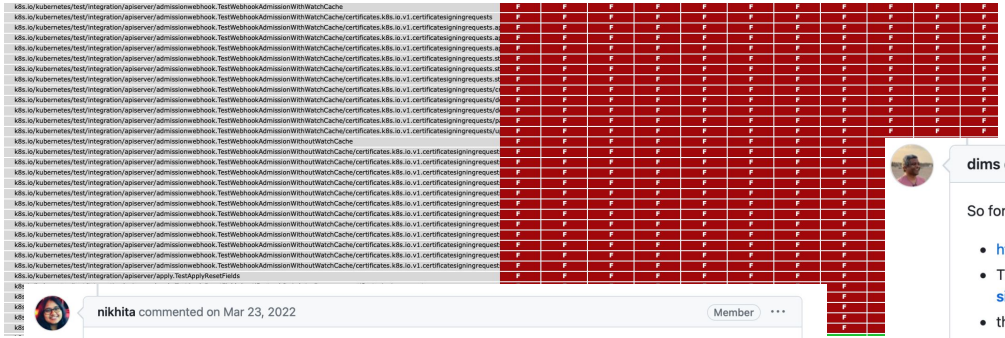


Fallacy #1: Timelines Are Reliable

These can affect the timelines of a release of the project.



Fallacy #1: Timelines Are Reliable



nikhita commented on Mar 23, 2022 — Member

> https://tip.golang.org/doc/go1.18#sha1

The failures for https://testgrid.k8s.io/sig-release-master-blocking#integration-master look like they are related to this.

Example below:

```
x509: cannot verify signature: insecure algorithm SHA1-RSA (temporarily override with GODEBUG=x509sha1=1)
```

dims commented on Apr 4, 2022 • edited — Member

So for folks following along:

- https://go-review.googlesource.com/c/go/+/394294/ needs to land first in golang master
- Then a backport CL has to be opened (by a bot?) for 1.18 branch which will get linked to **crypto/x509: reject SHA-1 signatures in Verify [1.18 backport]** golang/go#51852
- then a release has to be made by golang (looks like 1.18.1 which is expected in 3-4 days may not have the fix above)
- then we test the fix by dropping the temporary changes we made (https://cs.k8s.io/?q=x509sha1%3D1&i=nope&files=&excludeFiles=&repos=)
- then we cut v1.24.0 final

sounds right **@liggitt** ?

🎉 3

madhav 2 years ago

Looks like the Go 1.18.1 release is pushed back to 12th April: https://groups.google.com/g/golang-announce/c/vtbMjE04kPk/m/TY7AIrwSCgAJ?utm_medium=email&utm_source=footer
cc @palnabarun @james.laverack (edited)

dims 2 years ago

yep. just brought that to release folks attention on their call

dims 2 years ago

looks like this will make our release date slip for sure

Fallacy #1: Timelines Are Reliable

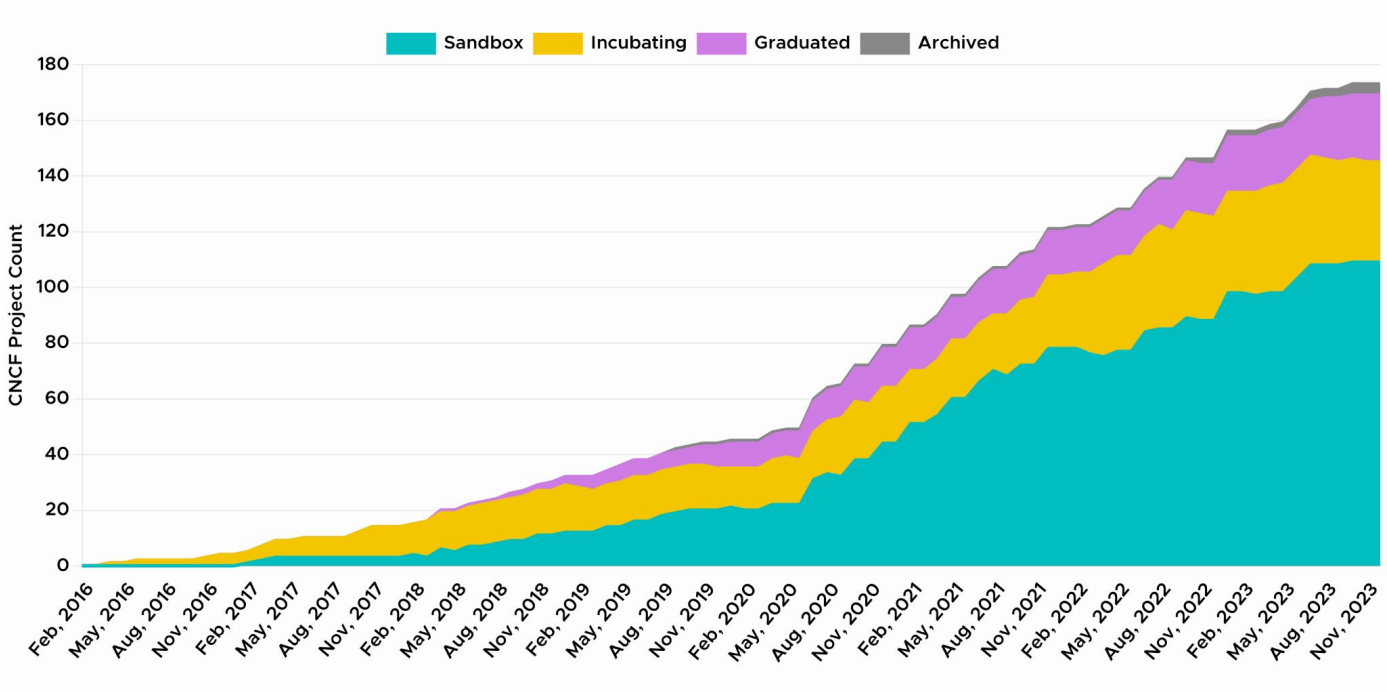
Timelines are **optimistic**

Fallacy #2: Feedback Loops Are Tight

Latency is zero: Ignorance of network latency, *and of the* ***packet loss it can cause***, *induces application- and transport-layer developers to allow* ***unbounded traffic***, *greatly increasing dropped packets and* ***wasting bandwidth***.

Fallacy #2: Feedback Loops Are Tight

Cloud Native Landscape is huge.



Fallacy #2: Feedback Loops Are Tight

It’s distributed too!

853 Members

Across 6 continents

178K Contributors

Fundamentally changing computing

Fallacy #2: Feedback Loops Are Tight

And the people maintaining are across a very diverse geography.

853 Members

Across 6 continents

178K Contributors

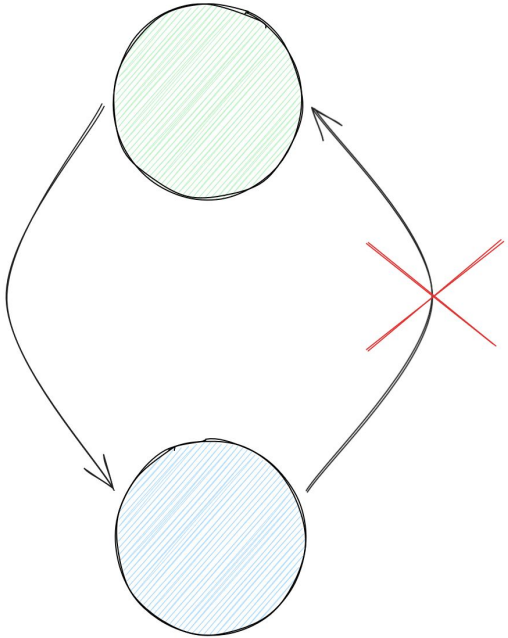
Fundamentally changing computing

Fallacy #2: Feedback Loops Are Tight

Fallacy #2: Feedback Loops Are Tight

Fallacy #2: Feedback Loops Are Tight

Feedback loops *can't* be tight in such a scenario



Fallacy #2: Feedback Loops Are Tight

Synchronous communication is nearly impossible

Fallacy #2: Feedback Loops Are Tight

Communicate asynchronously as much as possible to reduce overhead

Fallacy #2: Feedback Loops Are Tight

Discuss in a meeting *but* don't make decisions

Fallacy #2: Feedback Loops Are Tight

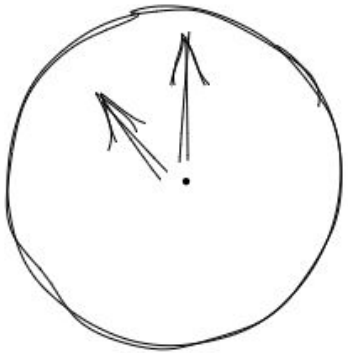
Make decisions lazily taking into account all opinions

Fallacy #3: Maintainer Bandwidth Is Infinite

Bandwidth is infinite: **Ignorance of bandwidth** limits on the part of traffic senders can **result in bottlenecks**.

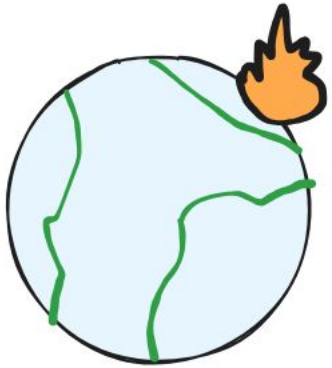
Fallacy #3: Maintainer Bandwidth Is Infinite

- A lack of bandwidth does not mean a lack of time.



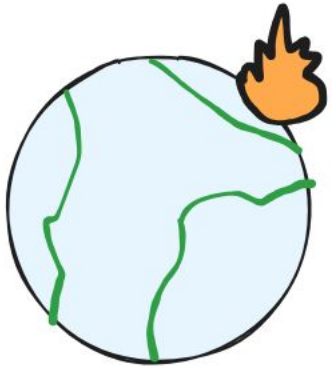
Fallacy #3: Maintainer Bandwidth Is Infinite

- A lack of bandwidth does not mean a lack of time.
- We unfortunately live in a world that is far from ideal and peaceful.



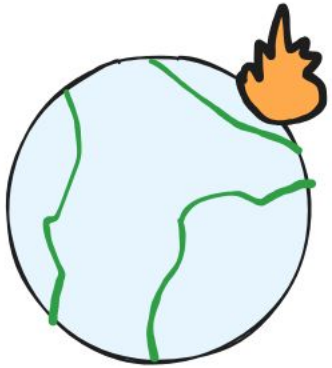
Fallacy #3: Maintainer Bandwidth Is Infinite

- A lack of bandwidth does not mean a lack of time.
- We unfortunately live in a world that is far from ideal and peaceful.
- As a result of which, our communities are going to be effected by it either directly or indirectly.



Fallacy #3: Maintainer Bandwidth Is Infinite

- A lack of bandwidth does not mean a lack of time.
- We unfortunately live in a world that is far from ideal and peaceful.
- As a result of which, our communities are going to be effected by it either directly or indirectly.
- Which is why in times like this we need to be extra empathetic when interacting with communities.



Fallacy #3: Maintainer Bandwidth Is Infinite

Maintainers love the projects they maintain and the community that comes with it, but when “life happens” this is a tried and tested formula for maintainer burnout.

Feeling of lack of control

+

A lack of empathy when spoken to

=

Sure shot recipe for burnout

Fallacy #3: Maintainer Bandwidth Is Infinite

It's always good to ask questions and request new things and all the niceness of open source, but be mindful when doing it. Help maintainers help you. Provide the fuel for the journey you're asking maintainers take on your behalf.

Fallacy #4: Commitments Don’t Change

Topology doesn’t change: **Changes** in network topology **can have effects** on both **bandwidth and latency issues**, and therefore can have similar problems.

Fallacy #4: Commitments Don’t Change

“With a sufficient number of users of an API, it does not matter what you promise in the contract: all observable behaviours of your system will be depended on by somebody.”

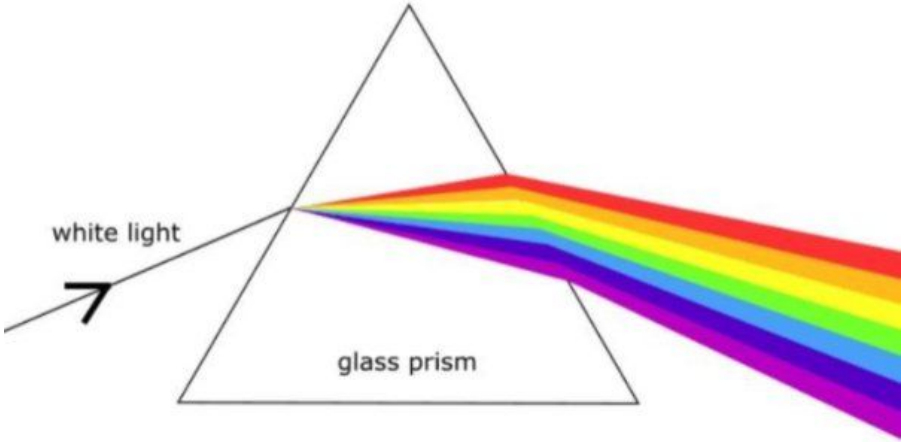
https://www.hyrumslaw.com/

Fallacy #4: Commitments Don’t Change

- As a project and its user base grows, the project starts getting used in ways that it never really was planned for.

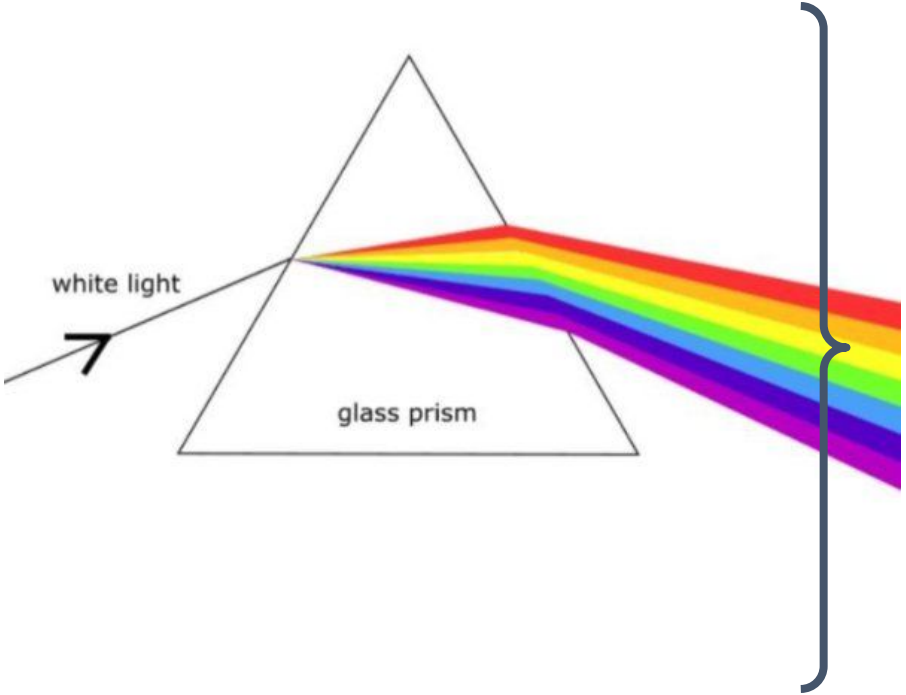
Fallacy #4: Commitments Don’t Change

- As a project and its user base grows, the project starts getting used in ways that it never really was planned for.
- This means the ways in which a project can break also starts becoming diverse.



Fallacy #4: Commitments Don’t Change

- As a project and its user base grows, the project starts getting used in ways that it never really was planned for.
- This means the ways in which a project can break also starts becoming diverse.
- But projects still want to accommodate for these cases to the best of their ability! In fact, if you’re using a project in novel ways, go tell your project maintainers!



Fallacy #4: Commitments Don’t Change

- However, sometimes - a project can go into survival, firefighting mode, optimizing for maximum compatibility and minimising blast radius.

Fallacy #4: Commitments Don’t Change

- However, sometimes - a project can go into survival, firefighting mode, optimizing for maximum compatibility and minimising blast radius.

Fallacy #4: Commitments Don’t Change

- However, sometimes - a project can go into survival, firefighting mode, optimizing for maximum compatibility and minimising blast radius.
- As a result of which, your niche breakage might not get fixed in any promised time frame, because remember - timelines are optimistic at best.

Fallacy #4: Commitments Don’t Change

- However, sometimes - a project can go into survival, firefighting mode, optimizing for maximum compatibility and minimising blast radius.
- As a result of which, your niche breakage might not get fixed in any promised time frame, because remember - timelines are optimistic at best.
- If you REALLY want it fixed, lend a helping hand, or maybe help put out the fire!

Fallacy #4: Commitments Don’t Change

KubeCon CloudNativeCon North America 2023

Wildfires, Firefighters and Sustainability - Learnings from Mitigating Kubernetes Fires in the Community - Nabarun Pal & Madhav Jivrajani, VMware

Click here to add to My Schedule.

https://sched.co/1HyeH

Fallacy #5: Software Supply Chain Is Secure

The network is secure: Complacency regarding network security *results in being* ***blindsided by malicious users*** *and programs that continually adapt to security measures.*

Fallacy #5: Software Supply Chain Is Secure

Have you ever downloaded the Kubernetes source code archive?

`https://github.com/kubernetes/kubernetes/archive/refs/heads/@kubernetes.zip`

Fallacy #5: Software Supply Chain Is Secure

If not, you should try that once.

Fallacy #5: Software Supply Chain Is Secure

But don’t try it from the URL in the previous slides.

Fallacy #5: Software Supply Chain Is Secure

You might ask why?

Fallacy #5: Software Supply Chain Is Secure

Because that's a malicious payload

`https://github.com/kubernetes/kubernetes/archive/refs/heads/@kubernetes.zip`

Fallacy #5: Software Supply Chain Is Secure

KubeCon CloudNativeCon North America 2023

SECURITY HUB: Hacking the Kubernetes Secure Software Supply-chain with .zip Domains - John McBride, Opensauced & Sean McGinnis, AWS

Sean McGinnis

Software Engineer, AWS

Sean McGinnis is an engineer working on the Bottlerocket container OS.

https://sched.co/1SKZK

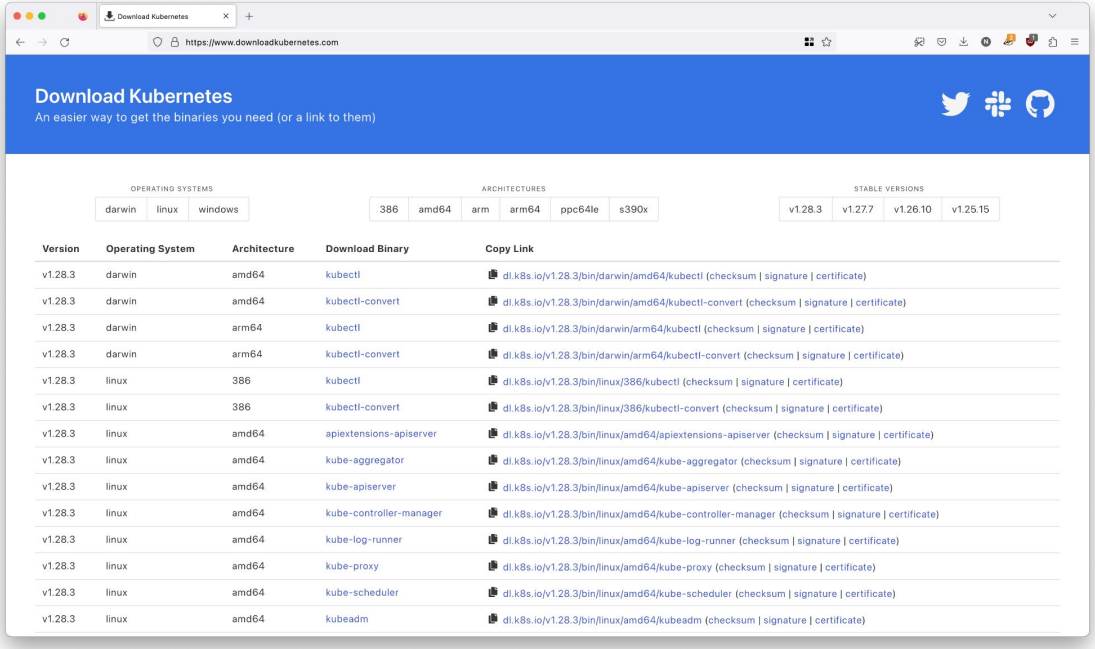
John McBride

Sr Software Engineer, OpenSauced

John is a Sr Software Engineer at OpenSauced where he works on open source insights and metric tooling. He's previously worked at AWS, VMware, and Pivotal. He maintains spf13/cobra in support of the CNCF ecosystem.

Fallacy #5: Software Supply Chain Is Secure

You should always download from verified sources.

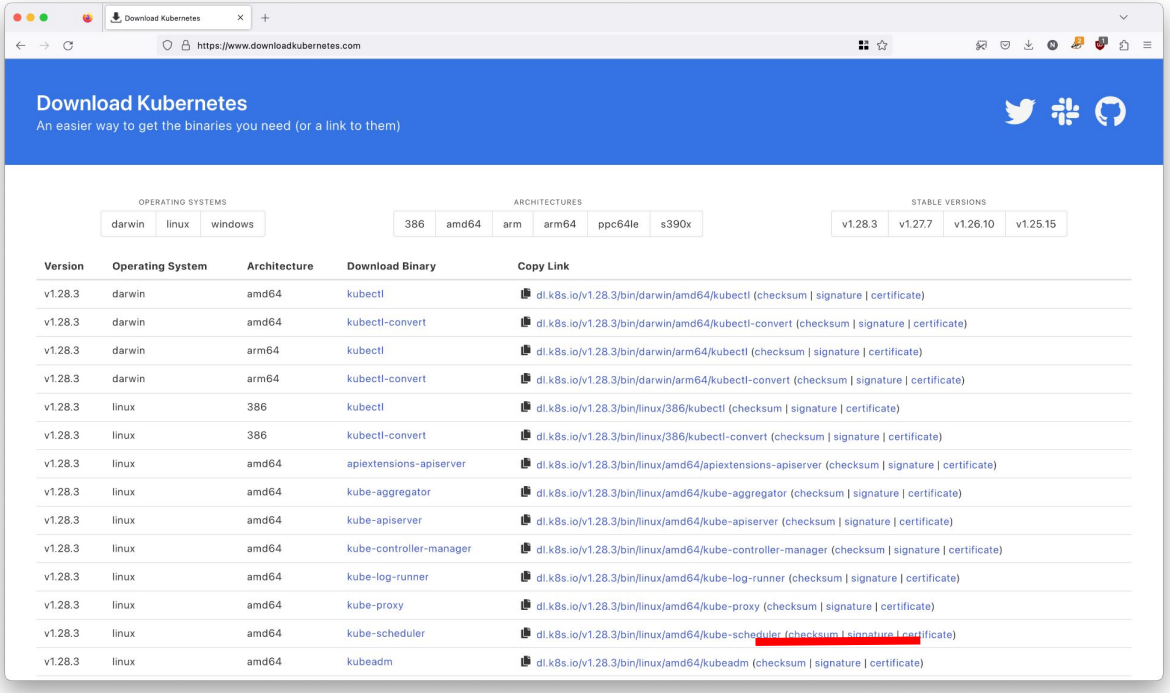


Fallacy #5: Software Supply Chain Is Secure

Even then, don’t believe me.

Fallacy #5: Software Supply Chain Is Secure

You should check the integrity of your artifacts.



https://kubernetes.io/docs/tasks/administer-cluster/verify-signed-artifacts/

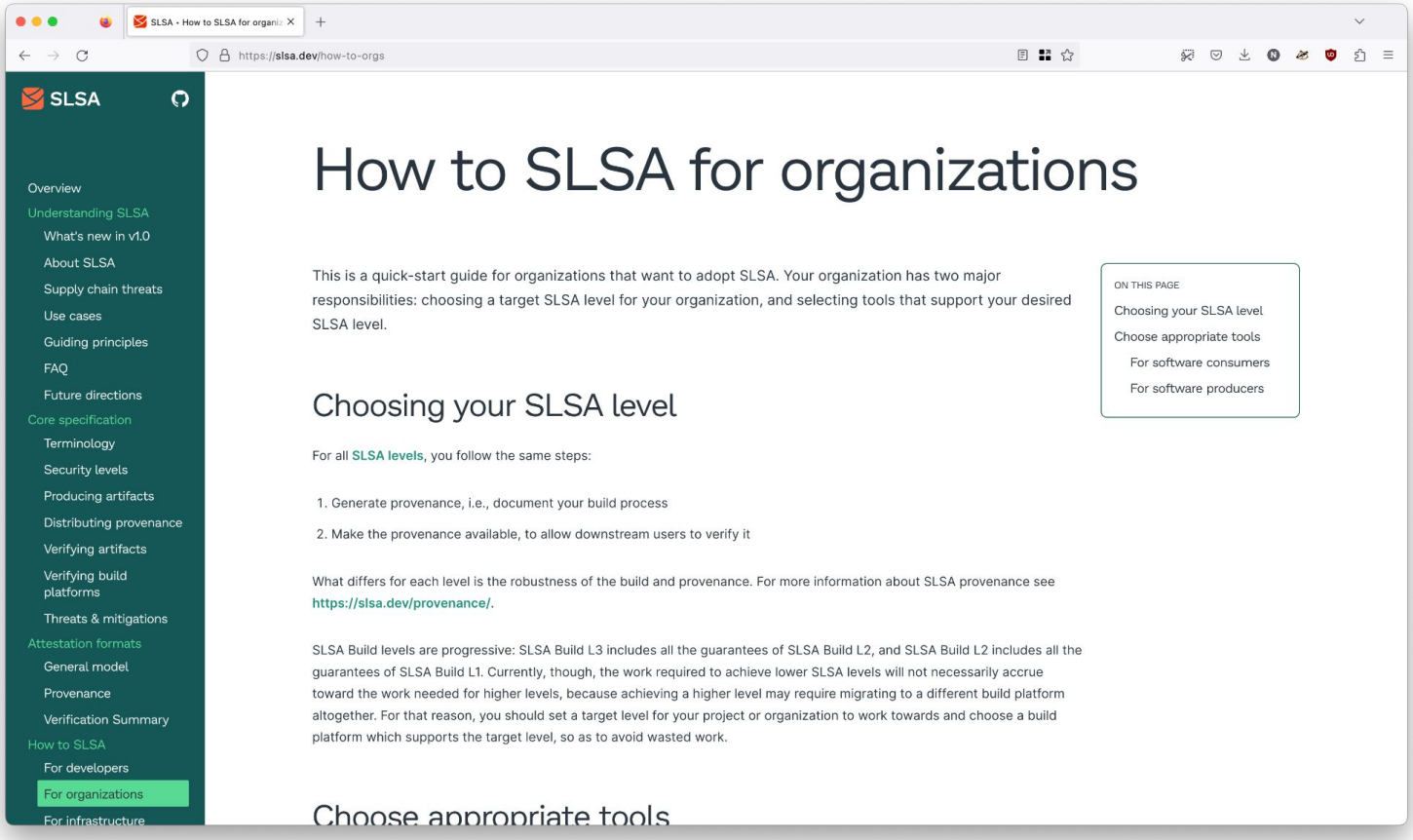
Fallacy #5: Software Supply Chain Is Secure

Fallacy: Software supply chain is secure.

Fallacy #5: Software Supply Chain Is Secure

You **NEED** to make it secure.

Fallacy #5: Software Supply Chain Is Secure



https://slsa.dev/get-started
https://slsa.dev/how-to-orgs

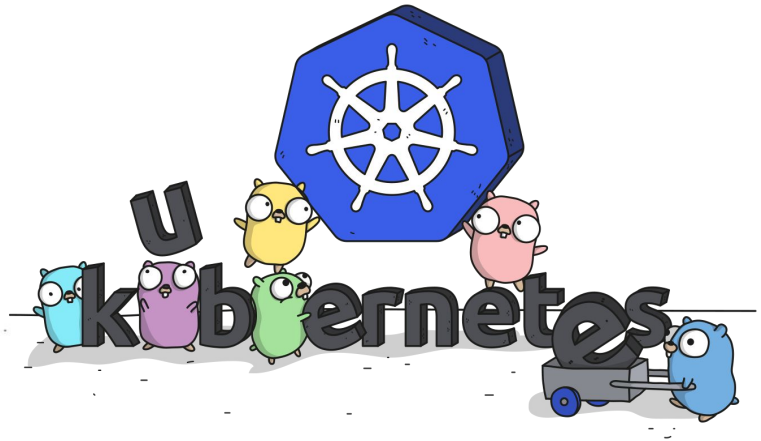
Fallacy #6: Compromise Is A Rarity And Not The Norm

There is one administrator: Multiple administrators, *as with subnets for rival companies,* ***may institute conflicting policies*** *of which senders of network traffic must be aware in order to complete their desired paths.*

Fallacy #6: Compromise Is A Rarity And Not The Norm

Maintaining large Open Source Projects is hard.

Fallacy #6: Compromise Is A Rarity And Not The Norm



#5 OSS project by developer activity*

#4 project by Pull Requests*

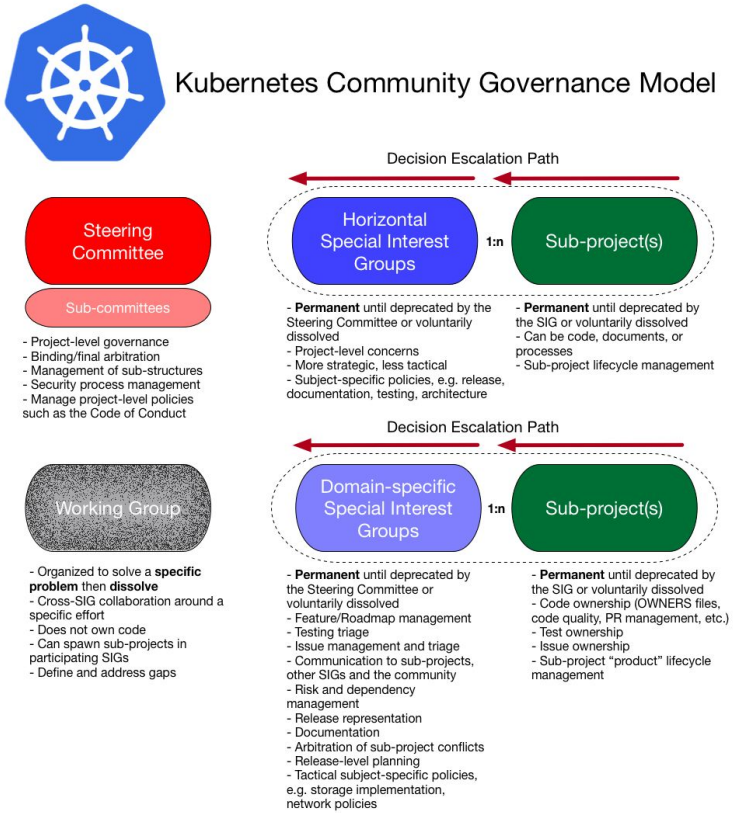
* Ref: CNCF Velocity Report

Community Stats (Oct 2023)	
Contributors	83,000~
Org Members	1800~
Repos	354
Community Groups	34

Source: devstats

Fallacy #6: Compromise Is A Rarity And Not The Norm

Often projects have multi-tiered governance structure



Fallacy #6: Compromise Is A Rarity And Not The Norm

Maintainers can have differing visions for the project.

Fallacy #6: Compromise Is A Rarity And Not The Norm

The incoherence shouldn’t affect the long term sustainability of the project.

Fallacy #6: Compromise Is A Rarity And Not The Norm

Kubernetes puts some checks and balances to make sure a community wide changes is adopted by a quorum.

Quorum

Quorum **to meet** is a ***majority of the fixed membership of the committee***.

Example:

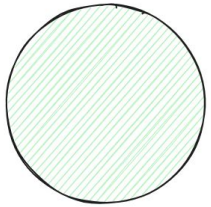
- 7 (members) / 2 = 3.5
- Round up to the nearest whole number (4)
- 4 members in attendance would be required to meet

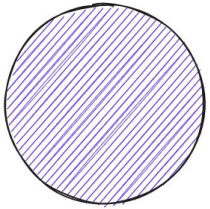
Quorum **to vote in a meeting** is a ***two-thirds supermajority of the fixed membership of the committee***.

Example:

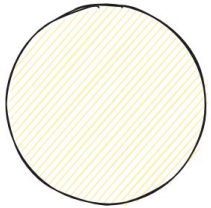
- 7 (members) / 3 = 2.333...
- 2.333... * 2 = 4.666...
- Round up to the nearest whole number (5)
- 5 members in attendance would be required to vote during a meeting

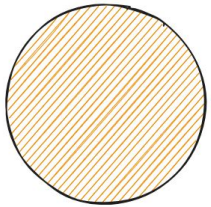
Fallacy #6: Compromise Is A Rarity And Not The Norm





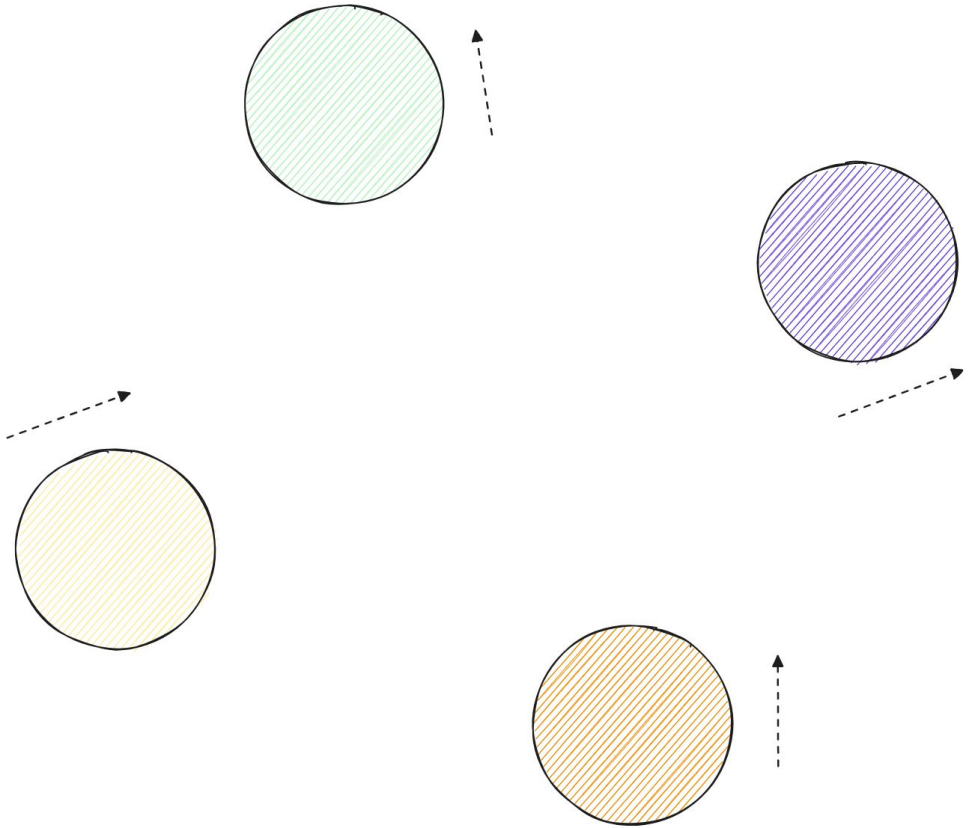
Similarly, other projects have multiple maintainers.





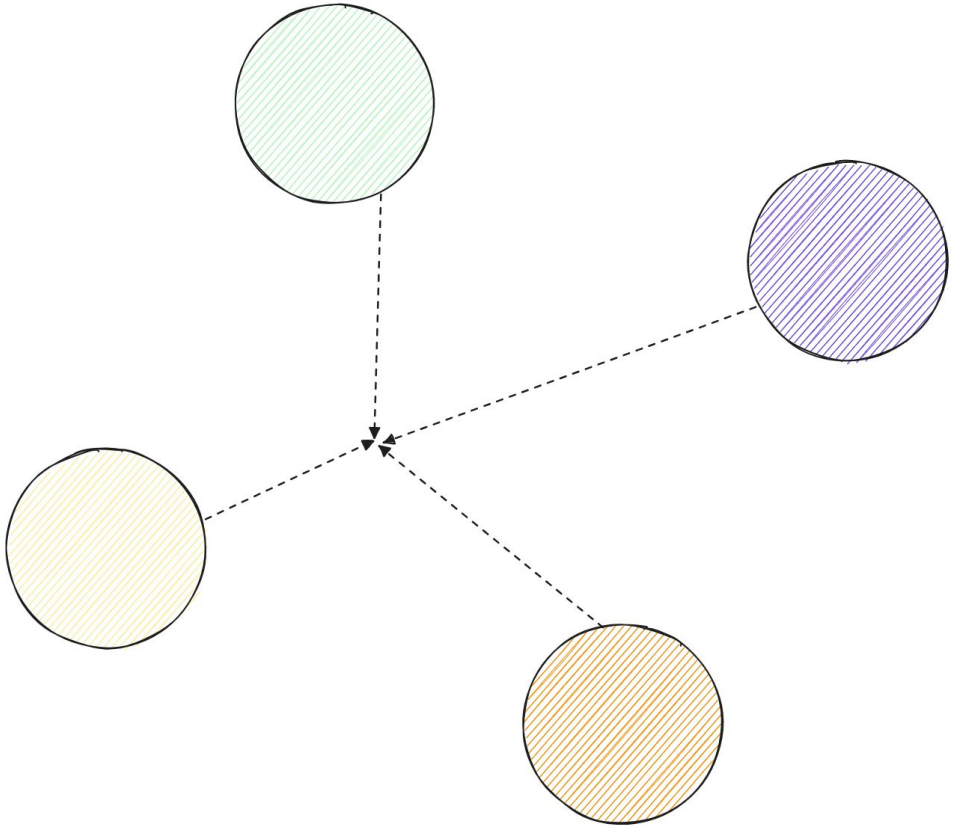
Fallacy #6: Compromise Is A Rarity And Not The Norm

Everyone has their own agenda.



Fallacy #6: Compromise Is A Rarity And Not The Norm

People compromise to come to a common conclusion.



Fallacy #7: Cost of Sustainably Onboarding Contributors Is Zero

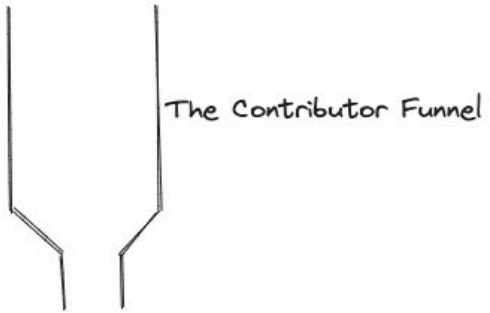
Transport cost is zero: **The "hidden" costs** of building and maintaining a network or subnet are **non-negligible** and must consequently be **noted in budgets to avoid vast shortfalls.**

Fallacy #7: Cost of Sustainably Onboarding Contributors Is Zero

- New contributors are the lifeblood of any open source community and are crucial from a sustainability point of view.

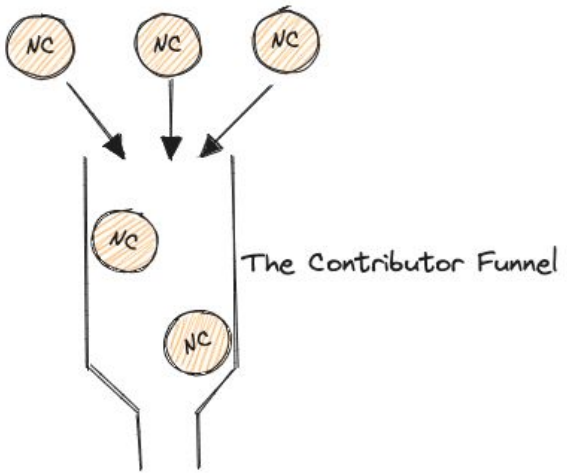
Fallacy #7: Cost of Sustainably Onboarding Contributors Is Zero

- New contributors are the lifeblood of any open source community and are crucial from a sustainability point of view.



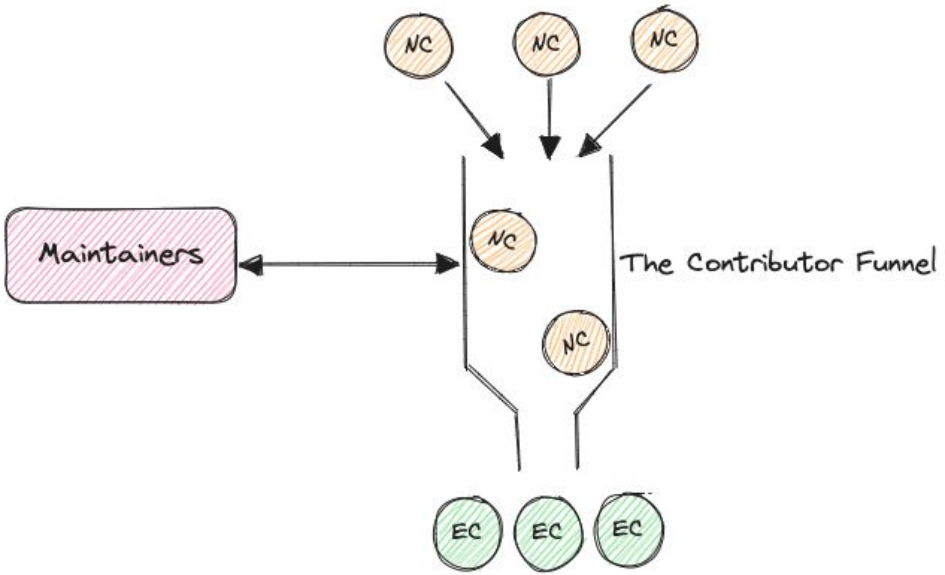
Fallacy #7: Cost of Sustainably Onboarding Contributors Is Zero

- New contributors are the lifeblood of any open source community and are crucial from a sustainability point of view.



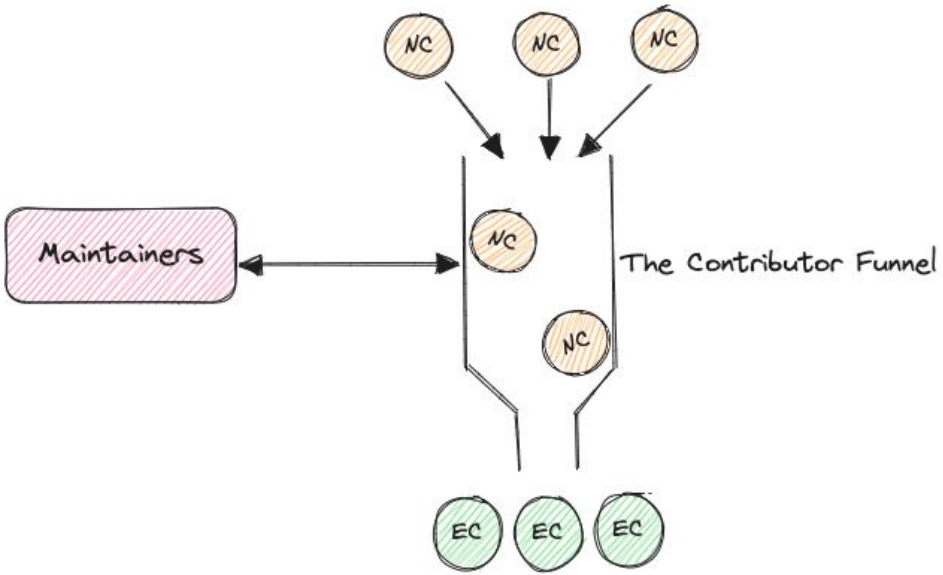
Fallacy #7: Cost of Sustainably Onboarding Contributors Is Zero

- New contributors are the lifeblood of any open source community and are crucial from a sustainability point of view.
- Maintainers help these new contributors get started to the best of their ability in hopes that they stick around and help out!



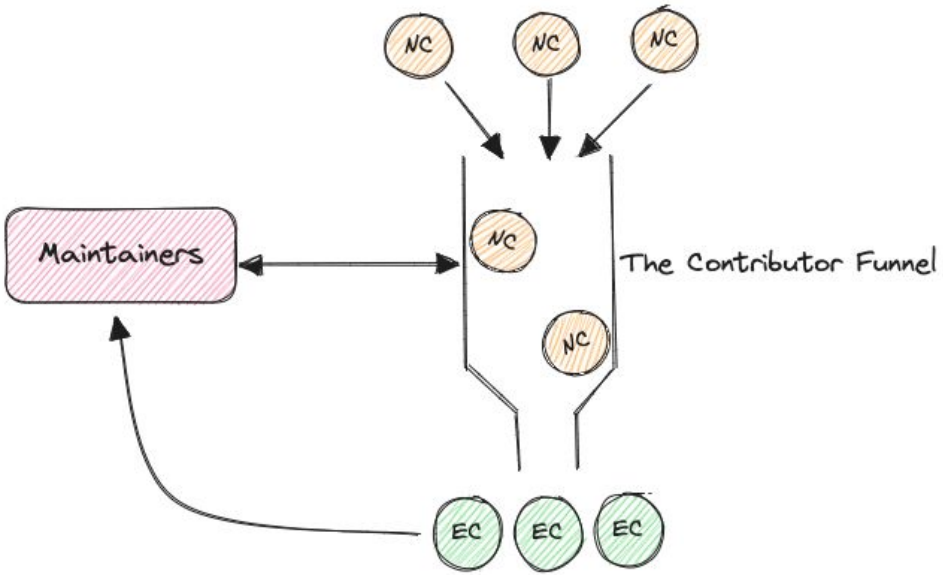
Fallacy #7: Cost of Sustainably Onboarding Contributors Is Zero

- New contributors are the life blood of any open source community and are crucial from a sustainability point of view.
- Maintainers help these new contributors get started to the best of their ability in hopes that they stick around and help out!
- New Contributors eventually become ”Episodic Contributors”.



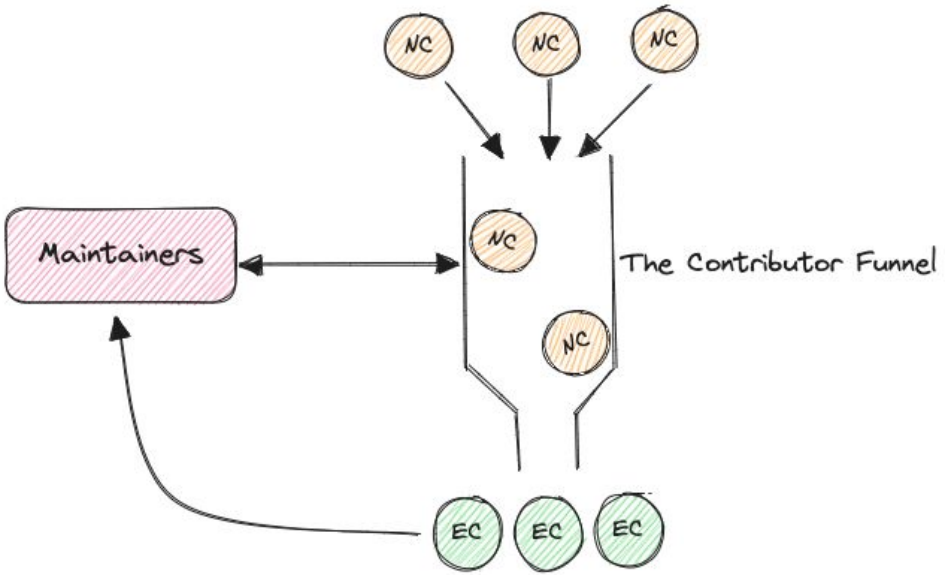
Fallacy #7: Cost of Sustainably Onboarding Contributors Is Zero

- New contributors are the lifeblood of any open source community and are crucial from a sustainability point of view.
- Maintainers help these new contributors get started to the best of their ability in hopes that they stick around and help out!
- New Contributors eventually become ”Episodic Contributors”.
- And ideally Episodic Contributors become maintainers and the cycle continues.



Fallacy #7: Cost of Sustainably Onboarding Contributors Is Zero

However, the cost of EC -> maintainers proves to be quite high as a project and community grows.



Fallacy #7: Cost of Sustainably Onboarding Contributors Is Zero

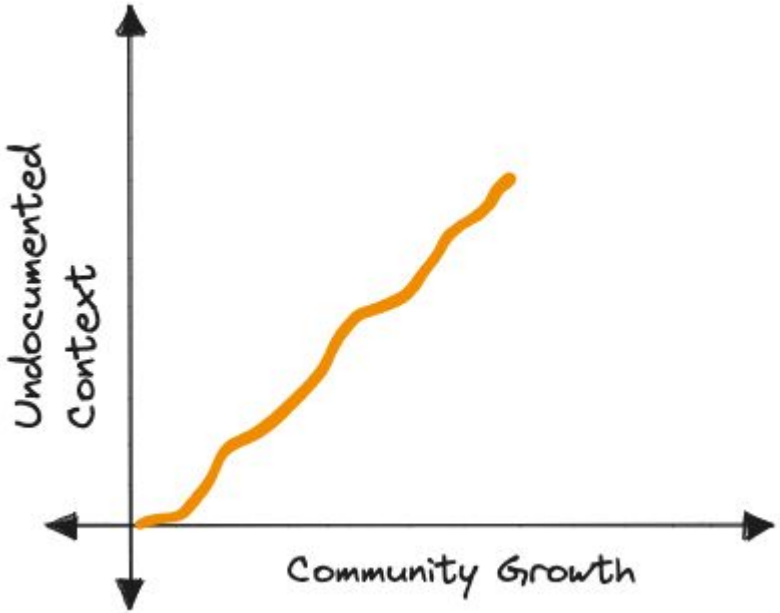
There are a few reasons for this:

1. As we saw – maintainer bandwidth is not infinite.

Fallacy #7: Cost of Sustainably Onboarding Contributors Is Zero

There are a few reasons for this:

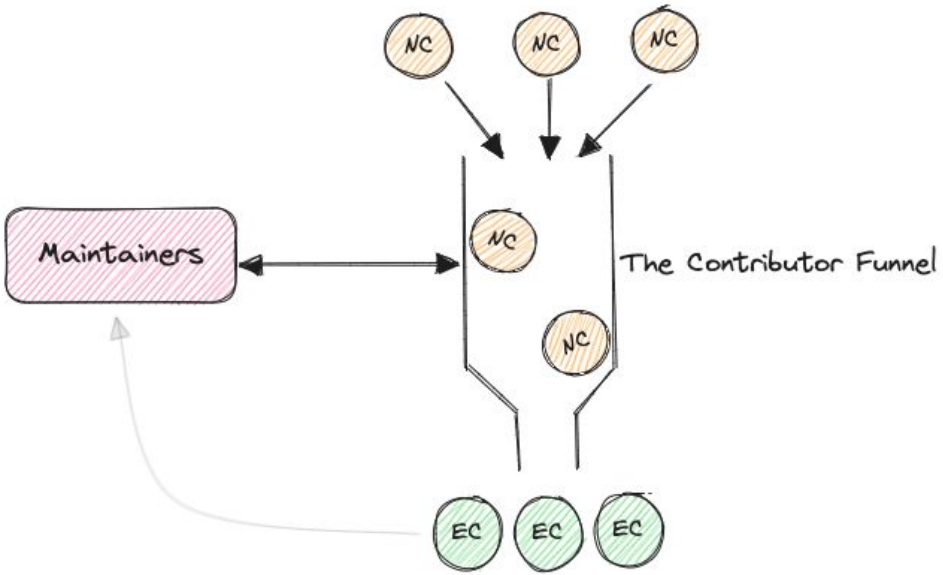
1. As we saw – maintainer bandwidth is not infinite.
2. Ownership of project areas gets hindered by undocumented context.



Fallacy #7: Cost of Sustainably Onboarding Contributors Is Zero

There are a few reasons for this:

1. As we saw – maintainer bandwidth is not infinite.
2. Ownership of project areas gets hindered by undocumented context.



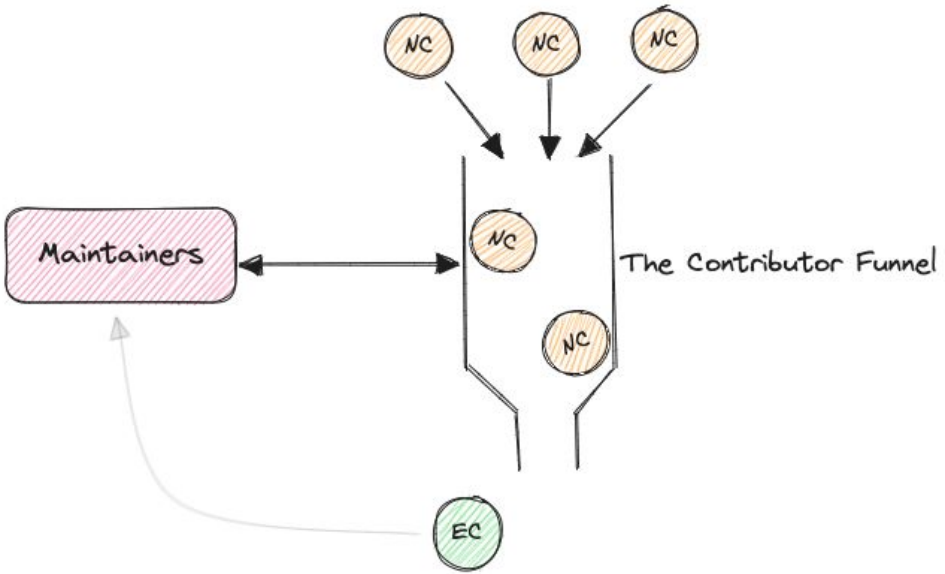
Fallacy #7: Cost of Sustainably Onboarding Contributors Is Zero

There are a few reasons for this:

1. As we saw – maintainer bandwidth is not infinite.
2. Ownership of project areas gets hindered by undocumented context.

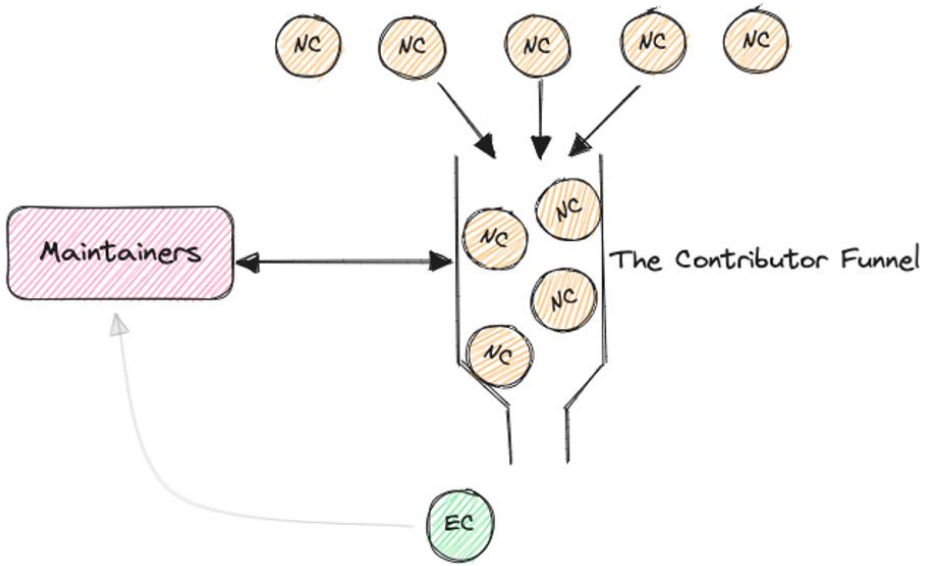
As a result of this:

- ECs leave.



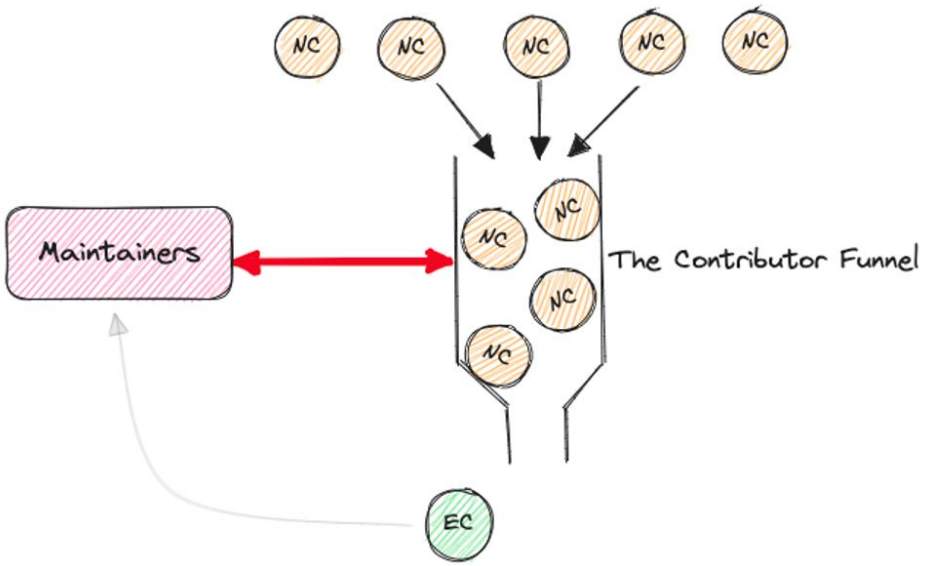
Fallacy #7: Cost of Sustainably Onboarding Contributors Is Zero

- But we still need new people, let's do more outreach!



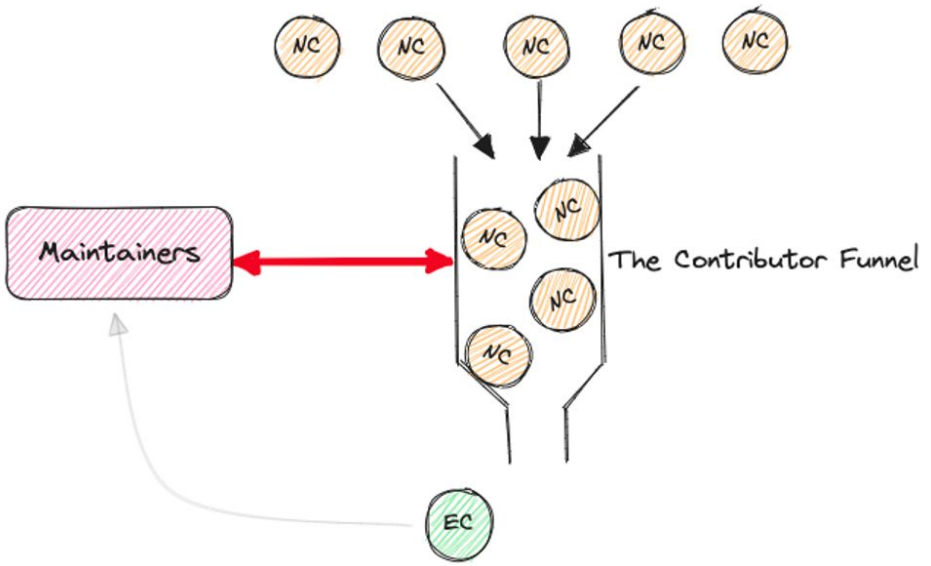
Fallacy #7: Cost of Sustainably Onboarding Contributors Is Zero

- But we still need new people, let's do more outreach!
- But the maintainer bandwidth is still constant.

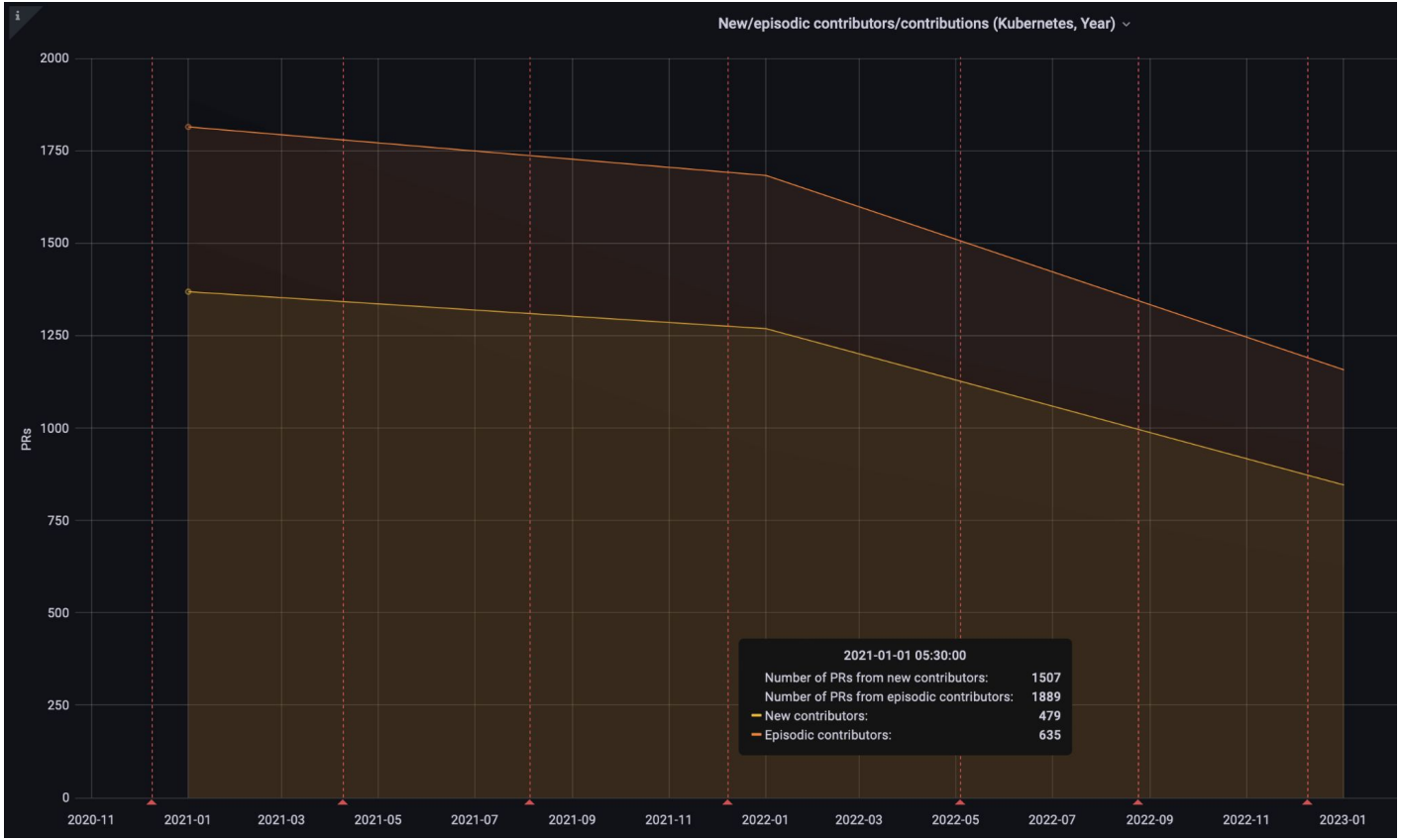


Fallacy #7: Cost of Sustainably Onboarding Contributors Is Zero

- But we still need new people, let's do more outreach!
- But the maintainer bandwidth is still constant.
- In a large project and community like Kubernetes, since the maintainer bandwidth is constant and often stretched thin, we don't have a mechanism for NCs to get the help they need!
- As a result of which, they drop off too.



Fallacy #7: Cost of Sustainably Onboarding Contributors Is Zero



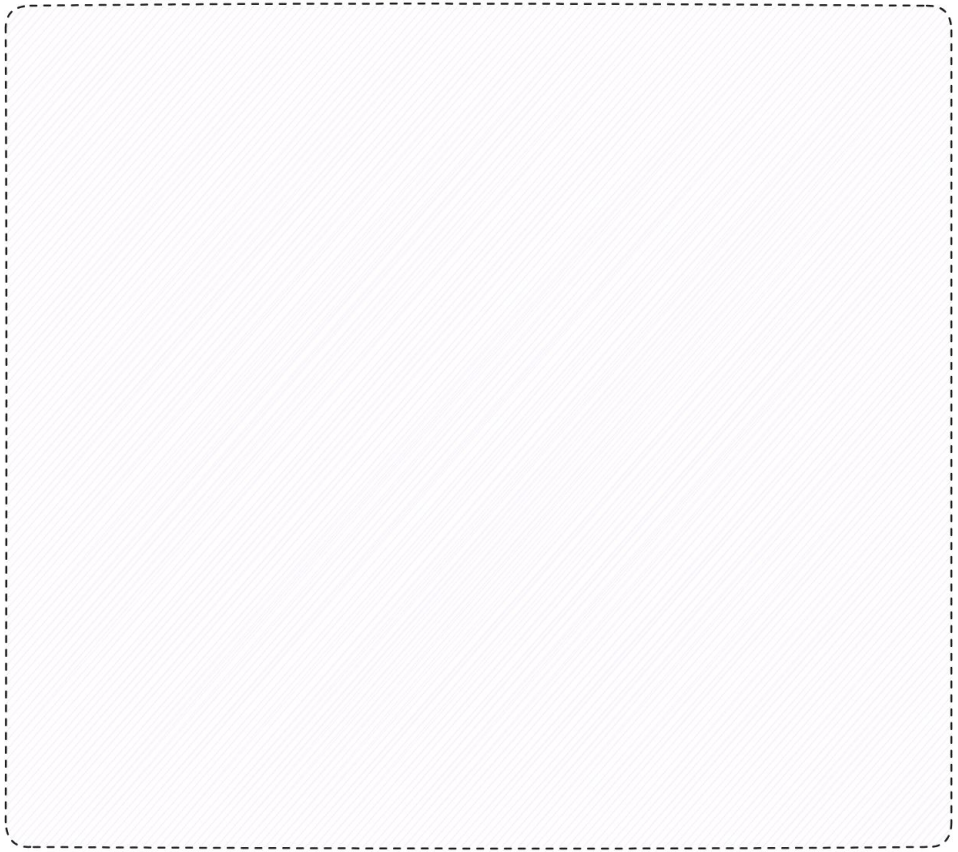
Source

Fallacy #8: Staffing Across Project Areas Is Homogenous

The network is homogenous: If a system **assumes a homogeneous network**, then it **can lead to the [...] problems** that result from the first three fallacies.

Fallacy #8: Staffing Across Project Areas Is Homogenous

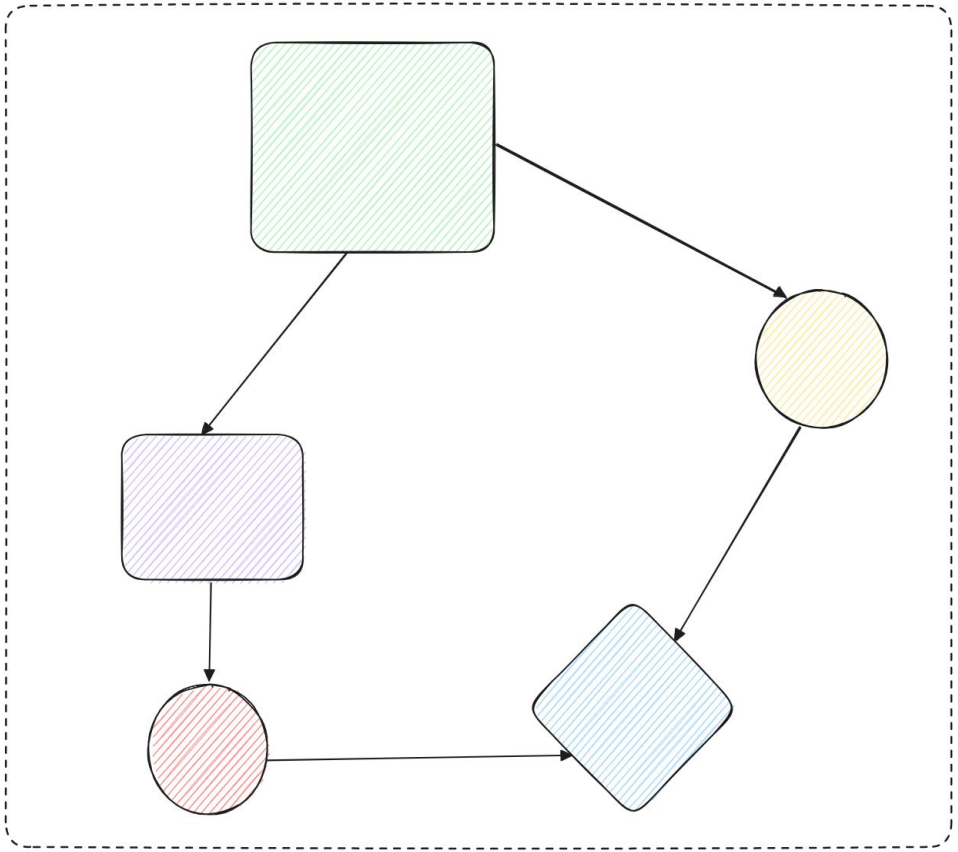
- A community can almost feel like a black box when you first interact with it.



Fallacy #8: Staffing Across Project Areas Is Homogenous

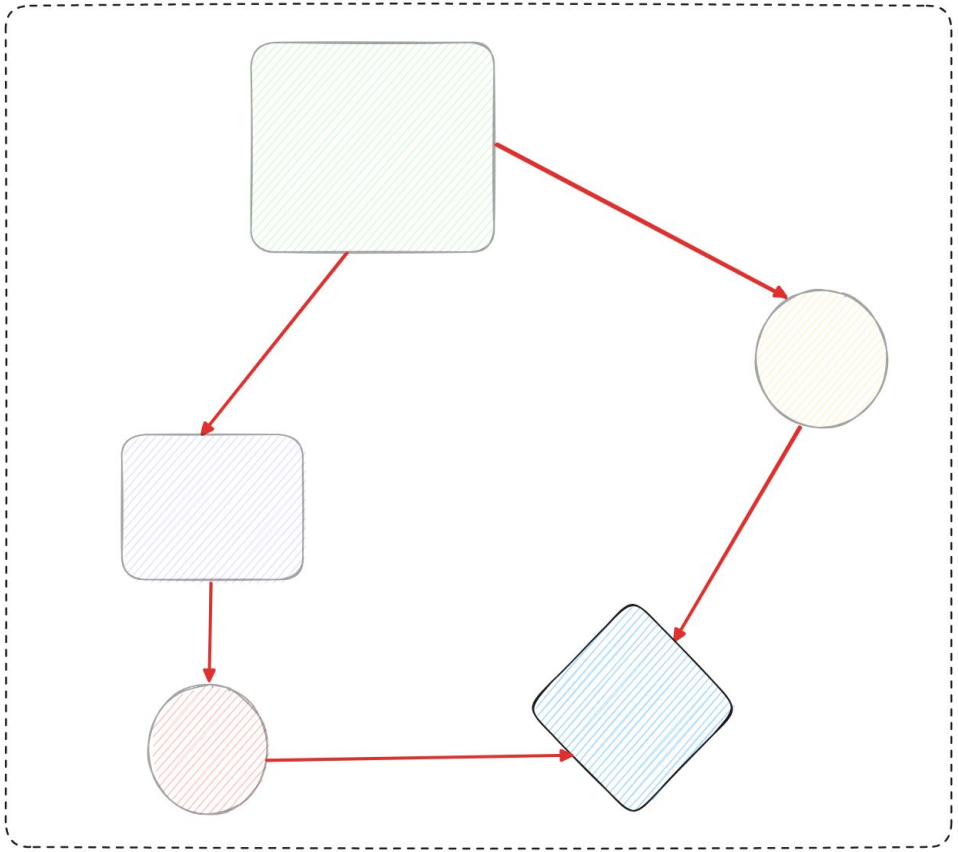
- A community can almost feel like a black box when you first interact with it.
- But the more time you spend, the different facets of it start emerging.

Open source communities are a web of socio-technical dependencies



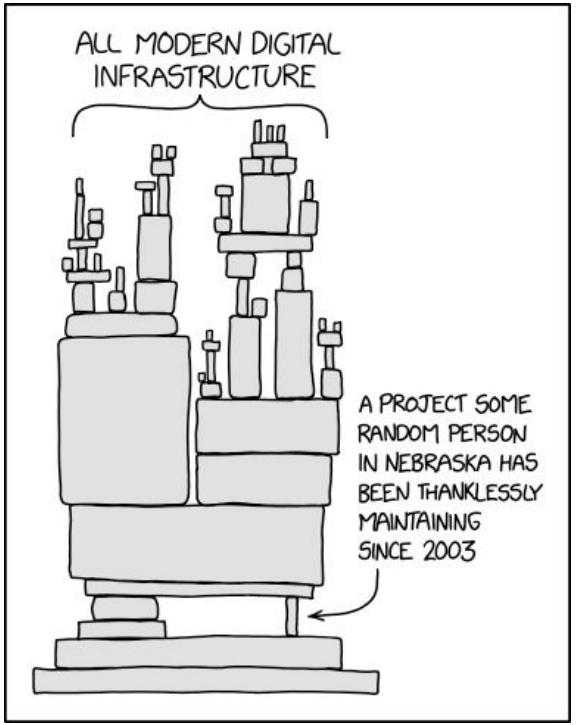
Fallacy #8: Staffing Across Project Areas Is Homogenous

- A community can almost feel like a black box when you first interact with it.
- But the more time you spend, the different facets of it start emerging.
- And soon it's not hard to see critical dependencies emerge.



Fallacy #8: Staffing Across Project Areas Is Homogenous

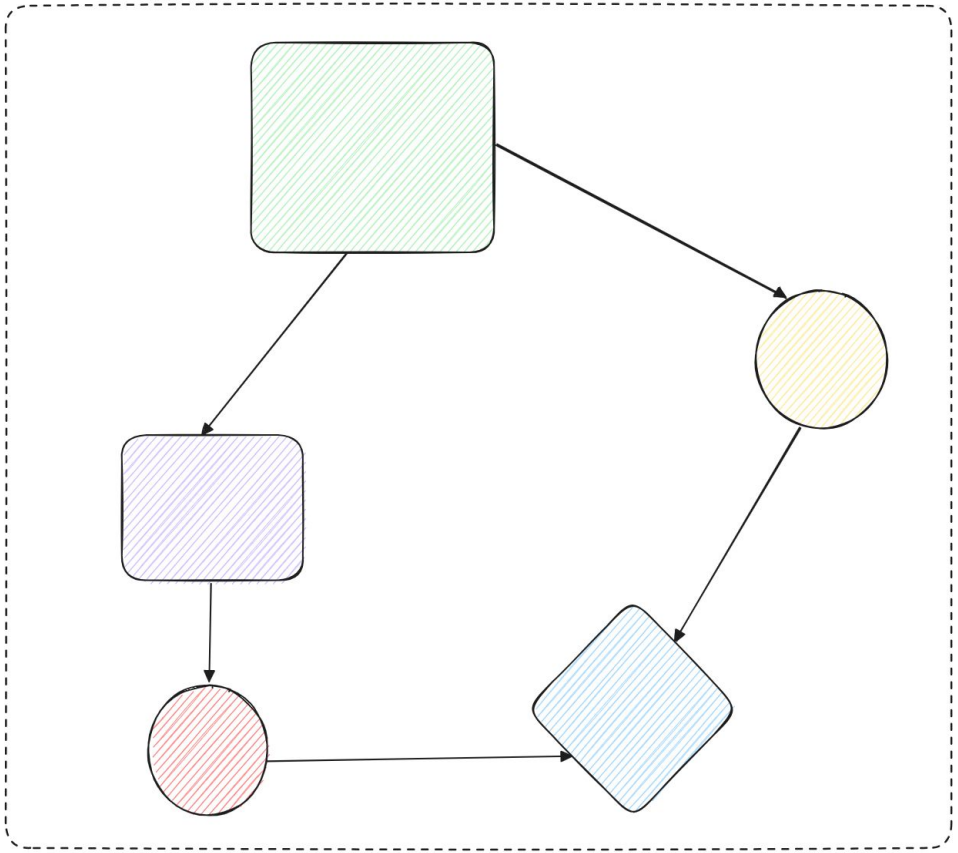
- A community can almost feel like a black box when you first interact with it.
- But the more time you spend, the different facets of it start emerging.
- And soon it's not hard to see critical dependencies emerge.



https://xkcd.com/2347/

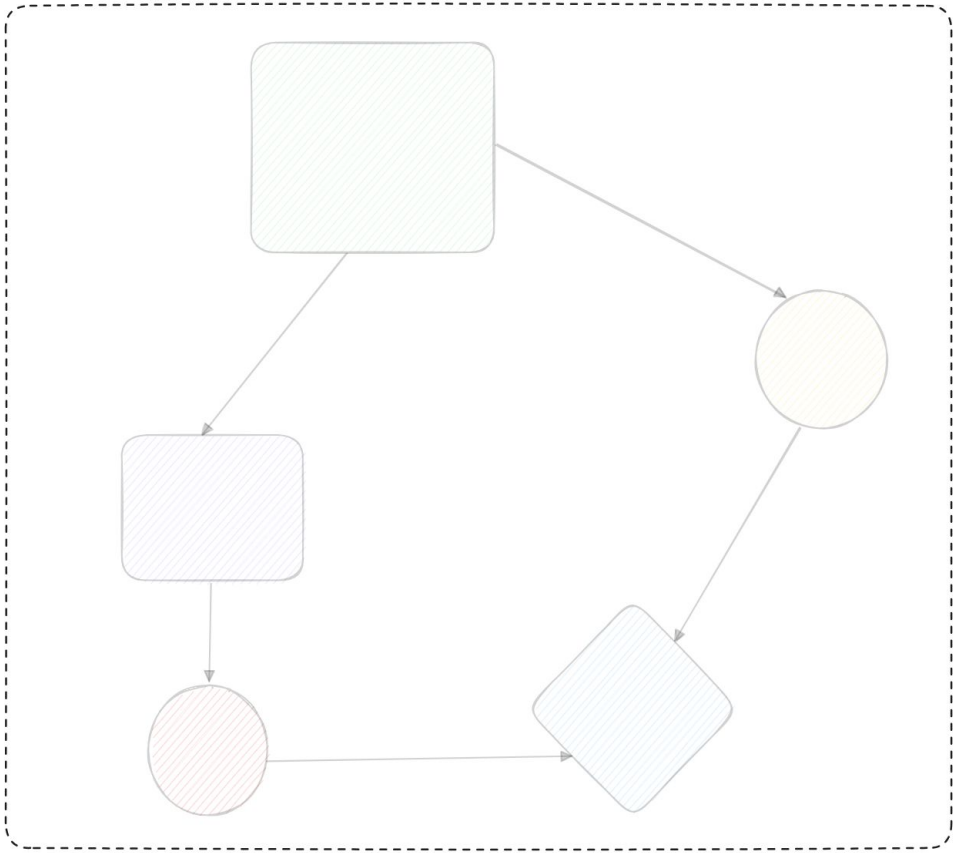
Fallacy #8: Staffing Across Project Areas Is Homogenous

- In a more general sense – not all areas of an open source project are staffed in proportion with their workload or critical dependence.



Fallacy #8: Staffing Across Project Areas Is Homogenous

- In a more general sense – not all areas of an open source project are staffed in proportion with their workload or critical dependence.
- So when the community still feels like a black box, it's easy to do quick math along the lines of “oh, there are so many contributors, why isn’t initiative **`xyz`** moving forward?”



Fallacy #8: Staffing Across Project Areas Is Homogenous

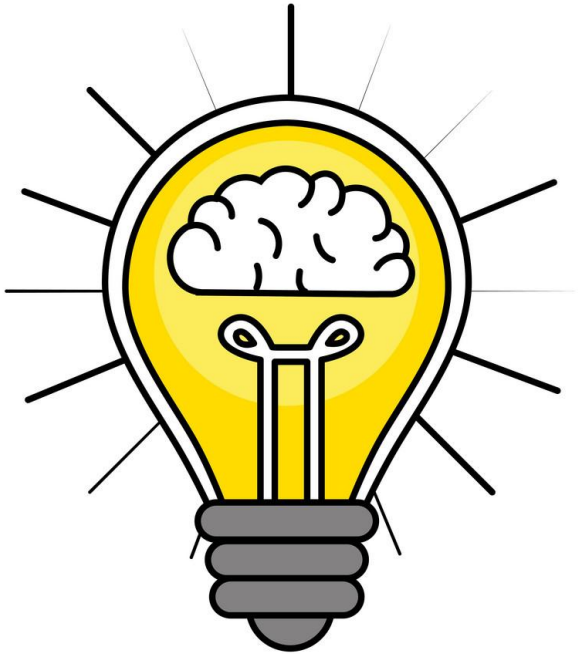
Understanding staffing needs of a project you rely on, is critical from your business continuity point of view.

Fallacy #8: Staffing Across Project Areas Is Homogenous

Sometimes funding contributors to work on areas you don't directly rely on can be the best thing you can do for the project and yourself.

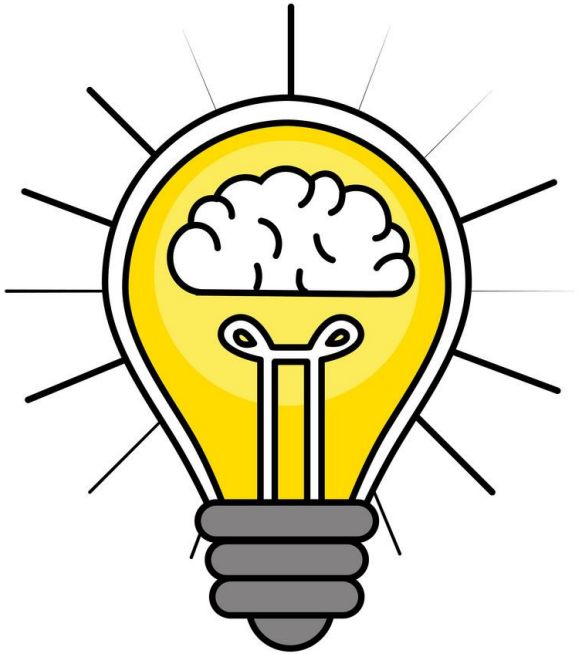
Concluding Thoughts

- Some of the fallacies have a solution



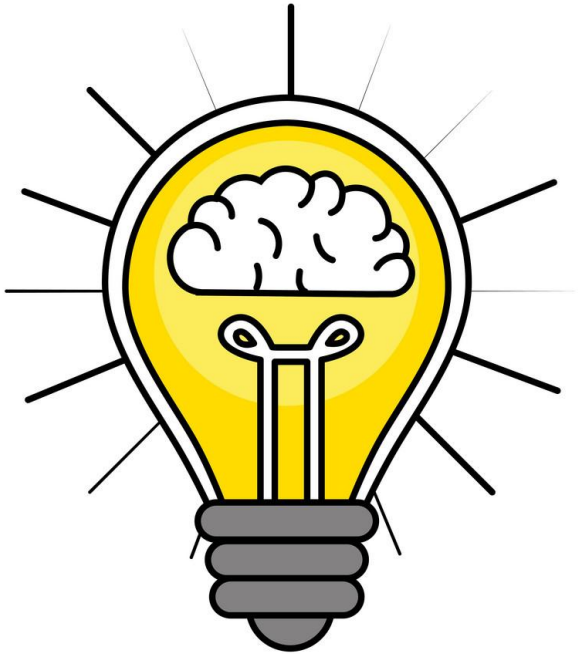
Concluding Thoughts

- Some of the fallacies have a solution
- Some may not!



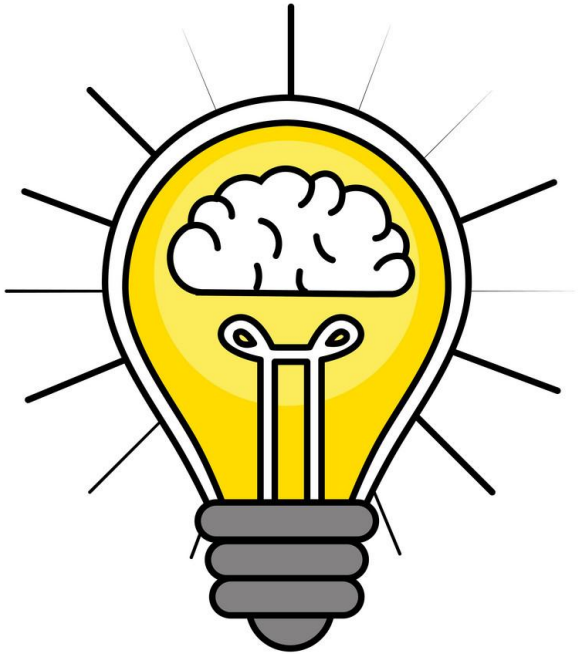
Concluding Thoughts

- Some of the fallacies have a solution
- Some may not!
- What is important is making sure communities are cognizant of the fallacies



Concluding Thoughts

- Some of the fallacies have a solution
- Some may not!
- What is important is making sure communities are cognizant of the fallacies
- This ensures a healthy contributor base



The Reality

Timelines are optimistic

The Reality

Timelines are optimistic

Prefer communicating asynchronously

The Reality

Timelines are optimistic

Prefer communicating asynchronously

Be extra empathetic and help maintainers help you

The Reality

Timelines are optimistic

Prefer communicating asynchronously

Be extra empathetic and help maintainers help you

If you use a project in unique ways, contribute your feedback and your skill!

The Reality

Timelines are optimistic

Prefer communicating asynchronously

Be extra empathetic and help maintainers help you

If you use a project in unique ways, contribute your feedback and your skill!

Make your software supply chain secure

The Reality

Timelines are optimistic

Prefer communicating asynchronously

Be extra empathetic and help maintainers help you

If you use a project in unique ways, contribute your feedback and your skill!

Make your software supply chain secure

Take into account maintainer incoherencies

The Reality

Timelines are optimistic

Prefer communicating asynchronously

Be extra empathetic and help maintainers help you

If you use a project in unique ways, contribute your feedback and your skill!

Make your software supply chain secure

Take into account maintainer incoherencies

With large communities, spend efforts on growing existing folks

The Reality

Timelines are optimistic

Prefer communicating asynchronously

Be extra empathetic and help maintainers help you

If you use a project in unique ways, contribute your feedback and your skill!

Make your software supply chain secure

Take into account maintainer incoherencies

With large communities, spend efforts on growing existing folks

Critical areas are the ones that are often understaffed

Meet the Kubernetes Contributors

KubeCon CloudNativeCon North America 2023

Thursday, November 9 • 12:00pm - 3:00pm

Meet the Kubernetes Contributor Community

Happening Now at W470AB!

https://sched.co/1T2qK

Kubernetes Steering Committee

KubeCon CloudNativeCon North America 2023

Thursday, November 9 • 2:55pm - 3:30pm

Learnings from Sustainably Steering the Kubernetes Project - Nabarun Pal, VMware & Bob Killen, Google

Nabarun Pal
Staff Software Engineer, VMware

Nabarun is a Staff Software Engineer at VMware, a maintainer of the Kubernetes project, an elected Kubernetes Steering Committee member, and a Kubernetes SIG Contributor Experience chair. He is a Release Manager for Kubernetes and has been the Kubernetes 1.21 Release Team Lead. Nabarun... Read More →

Bob Killen
OSS Program Manager, Google

Bob is a Program Manager at the Google Open Source Programs Office with a focus on Cloud Native computing. He serves the Kubernetes project as a Steering Committee member and chair of the Contributor Experience SIG. Bob comes from an academic background, spending 15 years at the University... Read More →

https://sched.co/1R2vZ

SIG Contributor Experience

KubeCon CloudNativeCon North America 2023

Inflections and Reflections from Kubernetes SIG ContribEx on Community Growth & Sustainability - Priyanka Saggu, SUSE; Madhav Jivrajani, VMware; Kaslin Fields, Google

Priyanka Saggu
Kubernetes Integration Engineer, SUSE

Priyanka Saggu, a Kubernetes Engineer at SUSE, has made significant contributions to Kubernetes project via Release, Testing, ContribEx, and CLI SIGs. She's the Release Lead for the ongoing Kubernetes v1.29 release cycle, Technical Lead for the Kubernetes's project SIG - ContribEx... Read More →

Madhav Jivrajani
Member of Technical Staff 2, VMware

Madhav loves to tinker with systems and is a Member of Technical Staff at VMware, working on open-source Kubernetes. Madhav is a maintainer and TL in the Kubernetes community and spends most of his time around areas of API-Machinery, Contributor Experience, Scalability and Archit... Read More →

Kaslin Fields
OSS K8s & GKE Developer Advocate, Google

Kaslin Fields is a Developer Advocate at Google Cloud, a Container enthusiast and creator of tech comics. She uses her knowledge of DevOps technologies and methodologies to help others as they enter the Cloud Native world. By creating comics about DevOps tech, she hopes to make learning... Read More →

https://sched.co/1R2ot

Please scan the QR Code above
to leave feedback on this session