



KubeCon



CloudNativeCon

Europe 2023





KubeCon



CloudNativeCon

Europe 2023


The Life and Times of a K8s Feature

Swati Sehgal, Red Hat
Francesco Romani, Red Hat



Agenda

1. Prerequisites
2. Communications
3. Contributor Ladder
4. Proposing a feature
5. Life of a feature: Alpha -> Beta -> GA (stable)
6. Contribution process

1. Special Interest Group (SIG)/Working Group (WG)
 - a. List of SIGs: <https://github.com/kubernetes/community/blob/master/sig-list.md>
 - b. Learn about SIGs and who's who of Kubernetes :
<https://github.com/kubernetes/community/blob/master/sigs.yaml>
 2. Kubernetes github organisation
 3. Kubernetes-sigs
-  **Tip:** Refer to <https://github.com/kubernetes/community> for more information

1. Slack: Register at slack.k8s.io
2. Mailing list
3. SIG Zoom Meetings



tip: If you are getting started, join [#sig-contribex](#) and [#kubernetes-novice](#) slack channels and attend [Kubernetes monthly community meeting](#).

Contributor Ladder

1. **Member:** active [contributor](#) in the community.
2. **Reviewer:** members most knowledgeable about areas of the code.
3. **Approver:** highly experienced reviewer. In charge of contributions acceptance approval.

Merge requirements

The PR must pass all automated tests

To merge a change, both **approve** and **lgtn** labels are needed

Approvers: can add **approve**, **lgtn**

Reviewers: can add **lgtn**

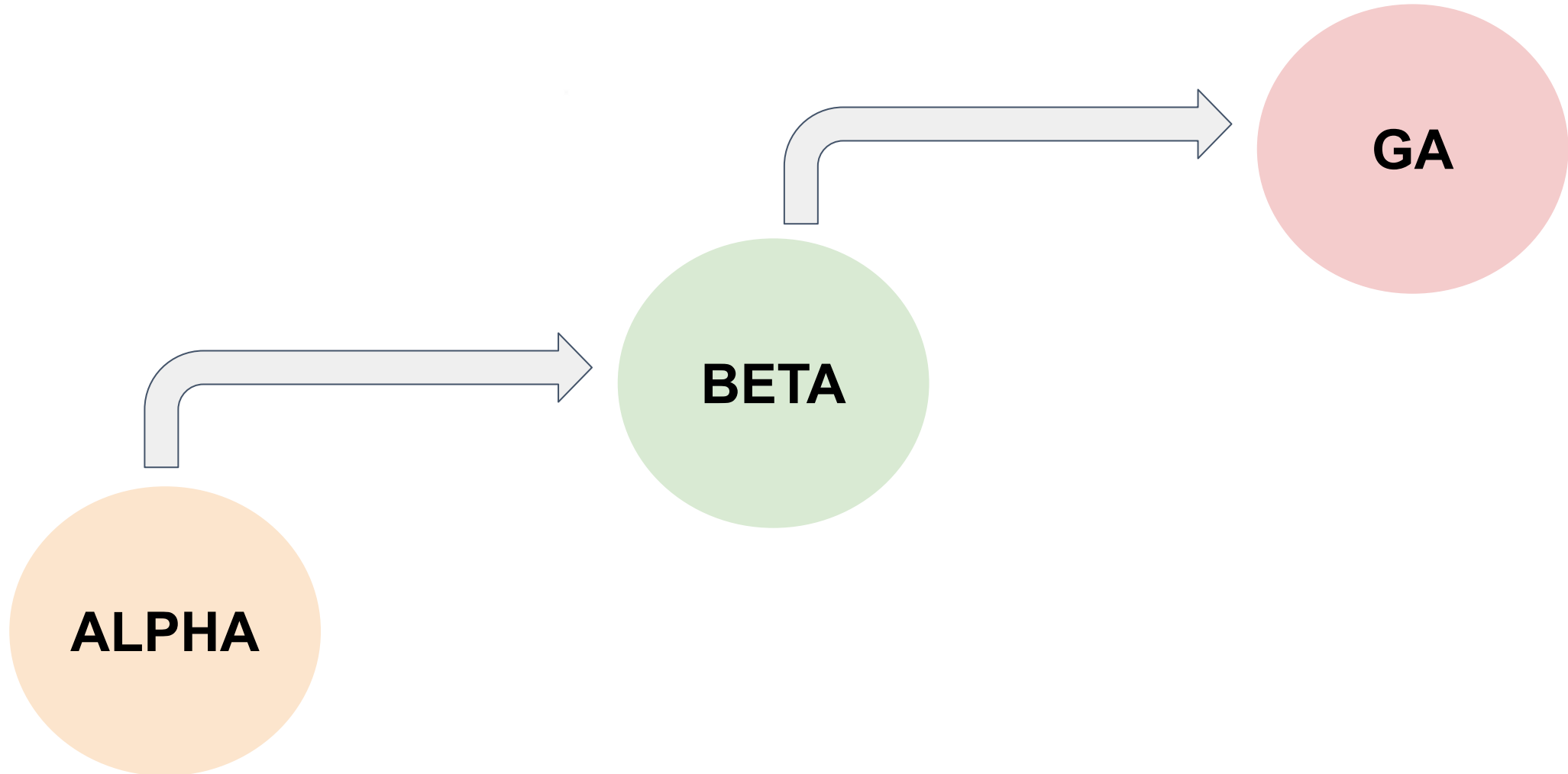
Reviewers and Approvers

You will need to coordinate with both one or more Approvers and one or more Reviewers.

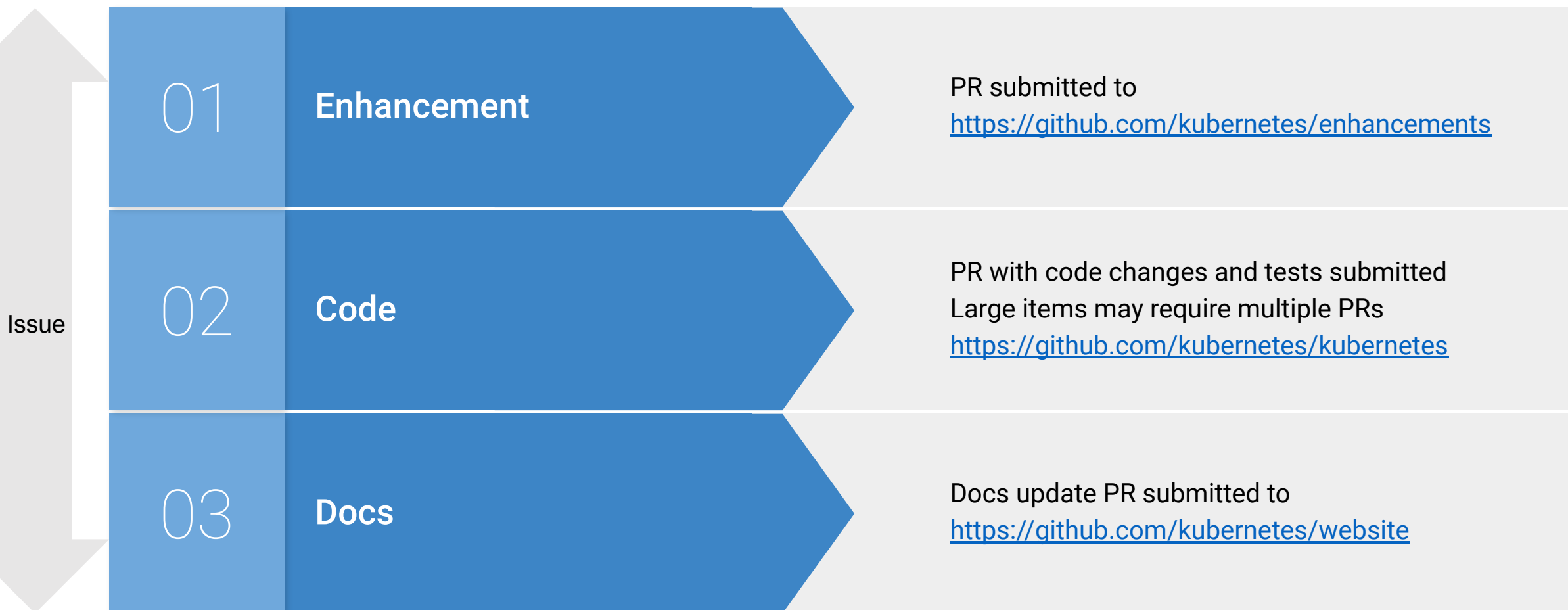
Approvers **usually** review once a change got **lgTM**. Their review is incremental

 **tip:** don't assume approver nor reviewer availability! Coordinate with them (slack messages, joining SIG meetings...)

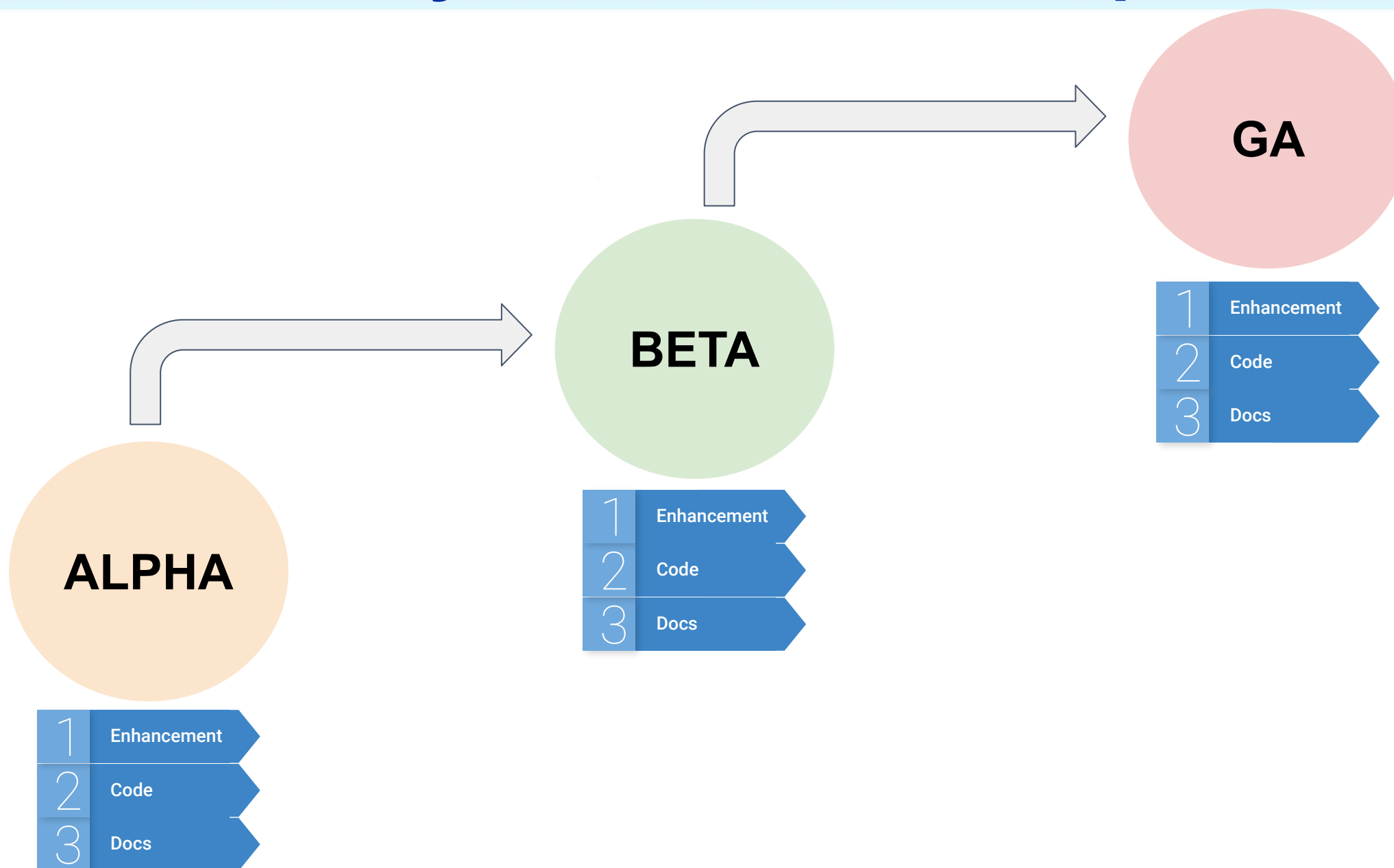
Feature Lifecycle



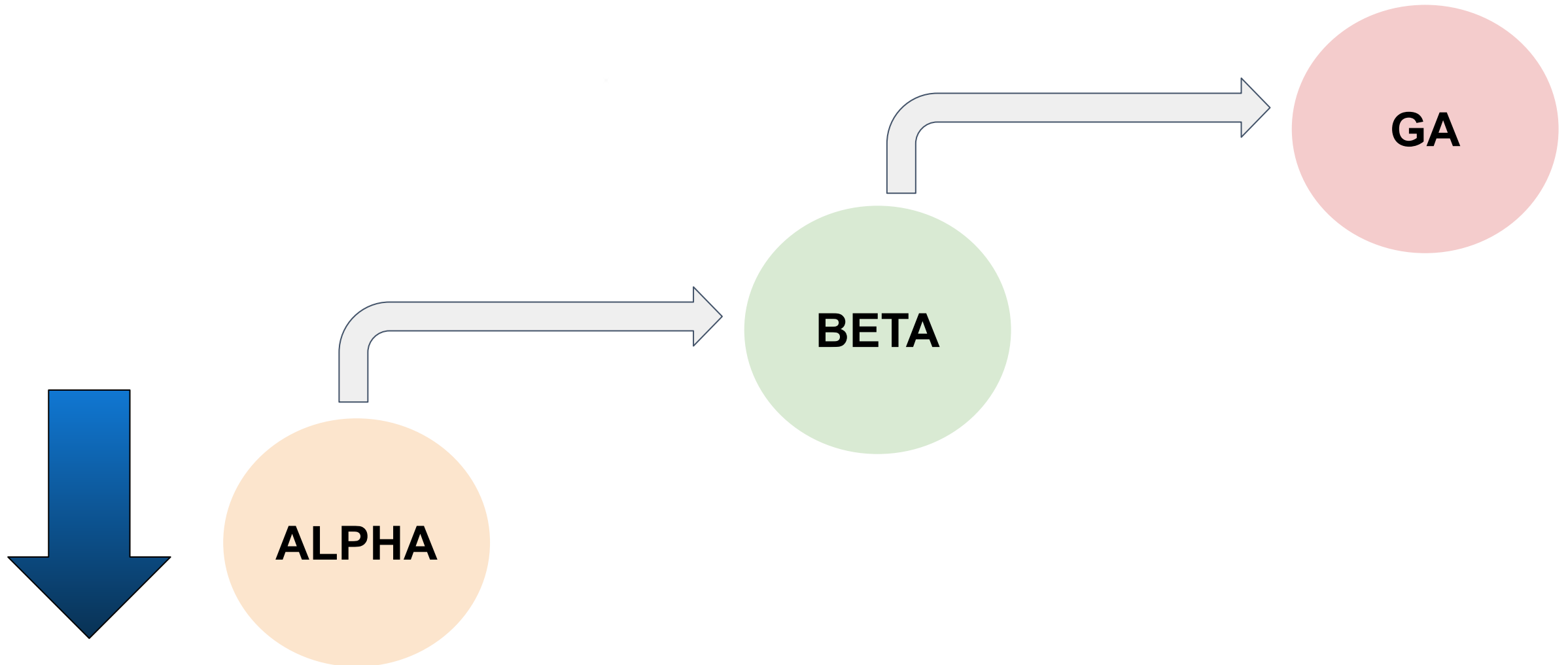
Process (simplified)



Feature Lifecycle & contribution process



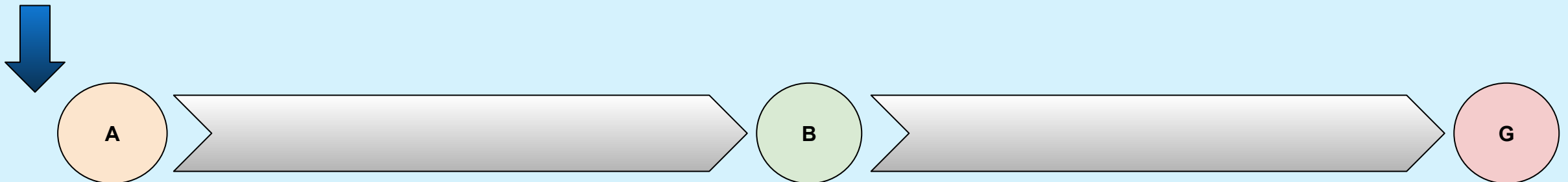
What happens in the pre-alpha stage?



What's a KEP?

A [Kubernetes Enhancement Proposal](#), or KEP, is the artifact supporting the process of enhancing kubernetes in a nontrivial way combining

- a feature/effort-tracking document
- a product requirements document
- a design document
- a document to evaluate production readiness of a feature



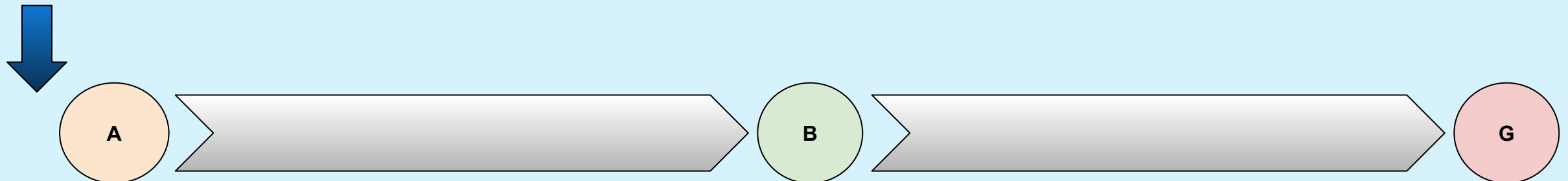
Good KEP practices

Start the KEP after having socialized the proposal

Gather consensus and ideas with less formal documents (google docs)

Review Production Readiness Questionnaire and keep in mind in the design process

 **tip:** iterate as quickly over the the KEP to get feedback and consensus

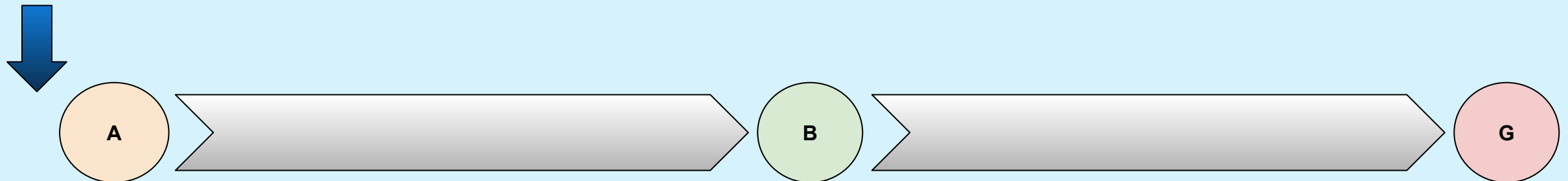


Scoping KEPs

Good KEPs have a very clearly defined scope

- a. KEPs can vary wildly in size - hard to predict the right size
- b. It's fine as long as scope and focus are clearly defined

It's completely fine and actually recommended to defer feature additions to further KEPs



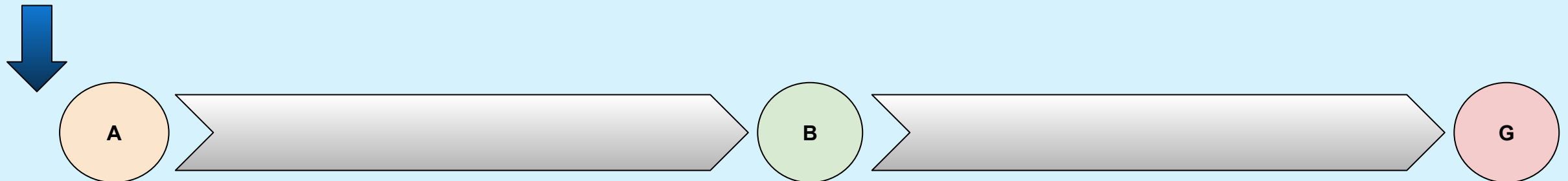
What to expect from the KEP review?

Feature Design, API and Production Readiness Review

The original feature scope can be narrowed down to a more defined subset

More KEPs may be required to fully implement the feature

The process will be even more iterative!



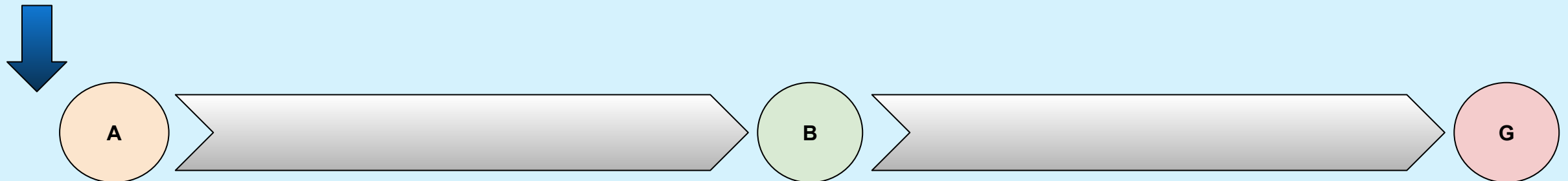
How did they do it?

Example feature 1: “[cpu manager extension to reject non SMT-aligned workload](#)”

Once gathered requirements, [share the idea on relevant SIG mailing list](#) and slack

Join the SIG meetings, [present your idea](#) and kickstart the discussion!

Gather feedback from the community, evaluate options. Sometimes [novel approaches](#) emerge!

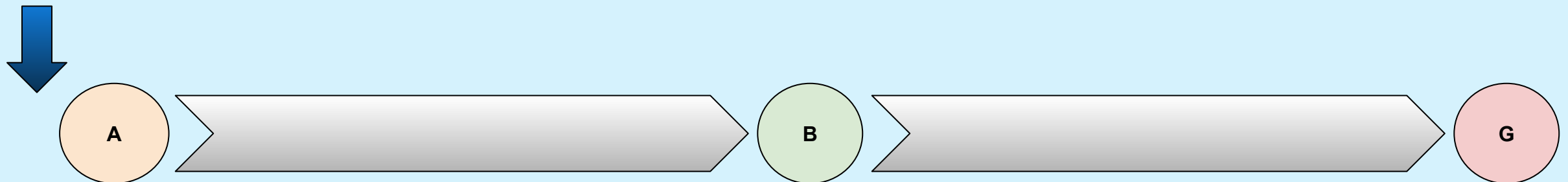


How did they do it?

Example feature 2: “[podresources endpoint to return allocatable resources](#)”

Proposed endpoint to learn about the resource allocation done by the kubelet

After discussion with sig-node, settled for: extension of the [existing monitoring API](#)



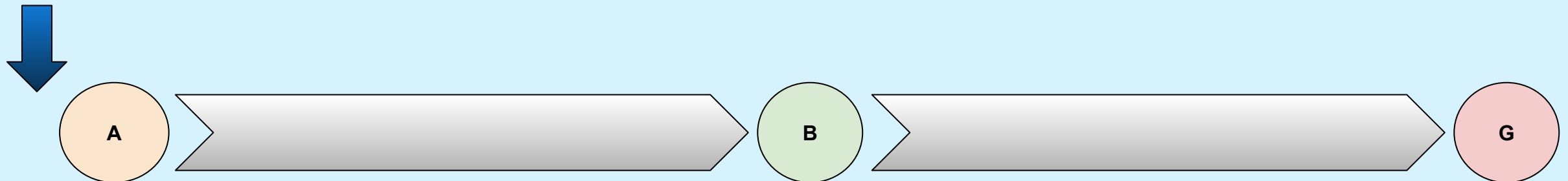
Alpha stage essentials

Audience: developers and expert users interested in giving early feedback on features

At least: the minimal viable subset of the feature

Suboptimal test coverage may still be accepted: aim for the maximum possible coverage given the circumstances!

There could be gaps or rough edges in general: gather feedback to fix them (chicken/egg issue)



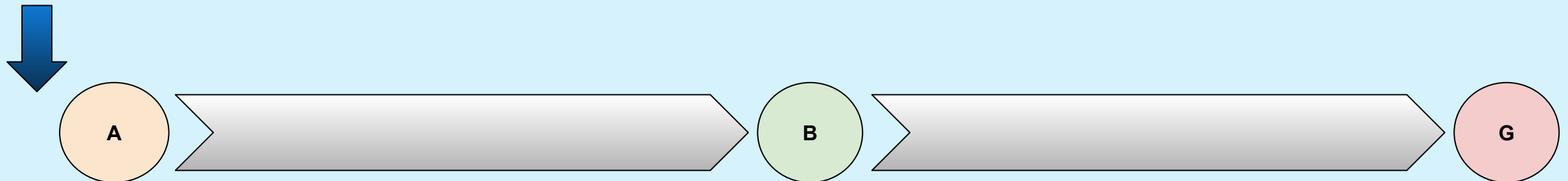
Alpha stage preparations

Time/resource allowing, it helps a lot to start preparing for:

[API review](#), if needed (or begin the process if you can!)

Production readiness review - required at each stage

- Initial focus on enablement, rollback
- The more you can prepare for the later stages, the better.

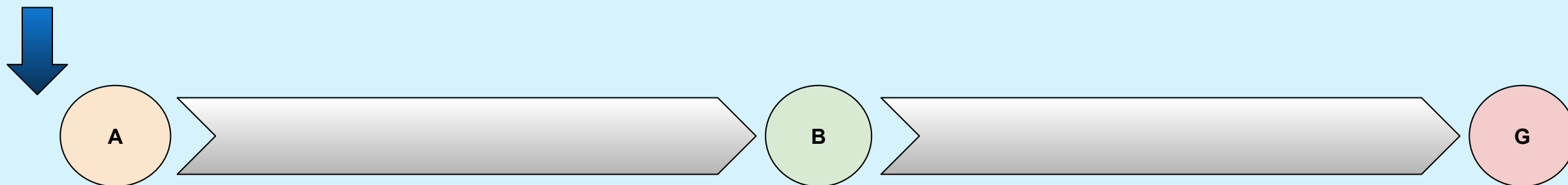


How did they do it?

Example feature 1: “[cpu manager extension to reject non SMT-aligned workload](#)”

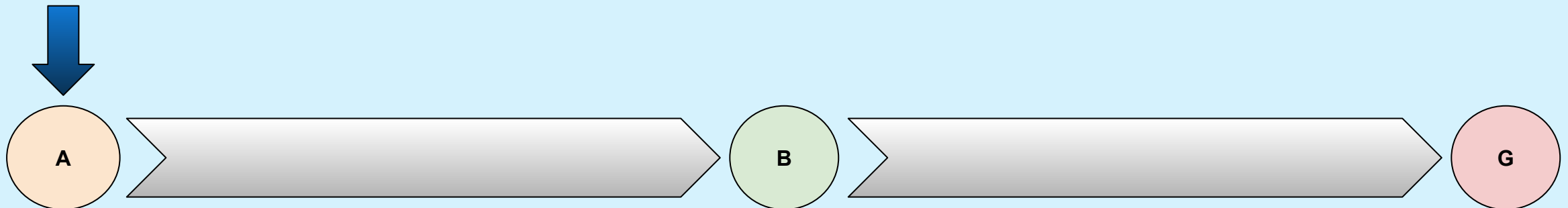
Added End-to-End early in the cycle!

Gain confidence in the feature



Alpha Stage

Celebrate! The feature is Alpha now! 🎉



What happens in the post-alpha stage?

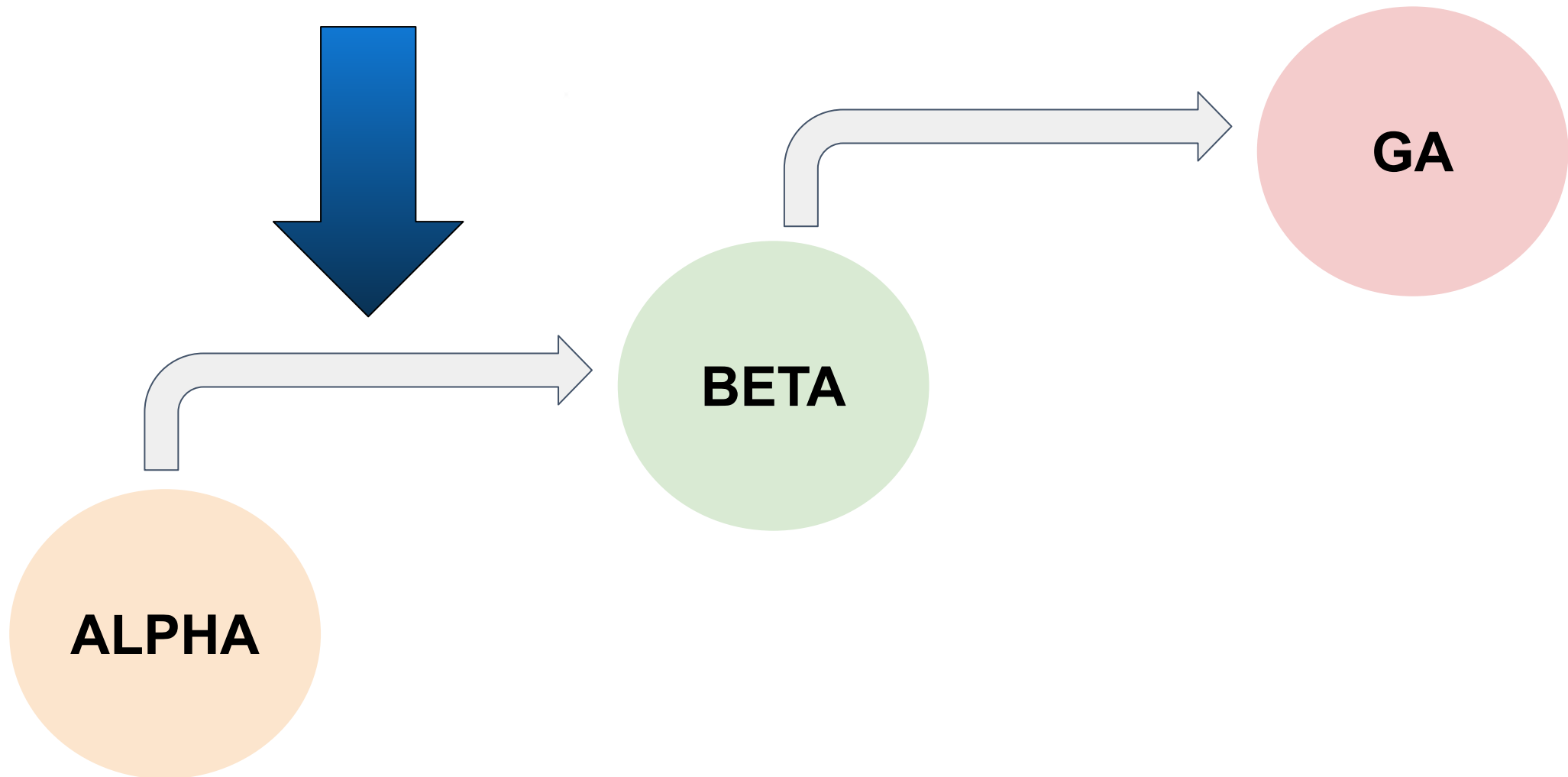


KubeCon



CloudNativeCon

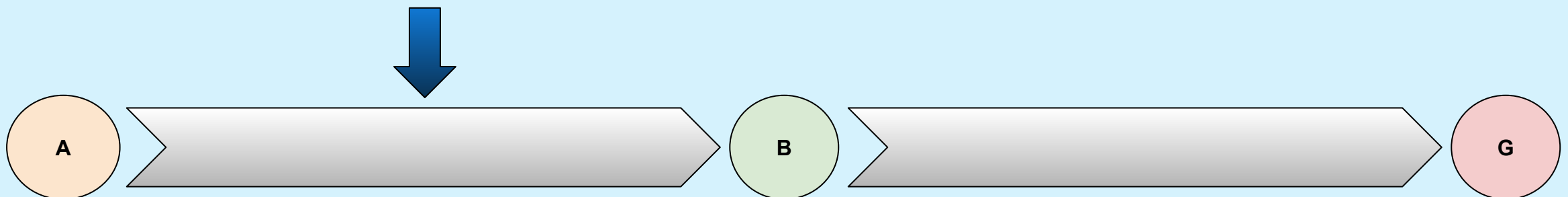
Europe 2023



Beta stage essentials

Audience: users interested in providing feedback on features

- [Feature Gate enabled by default](#)
- [Beta APIs are disabled by default](#) for APIs introduced after v1.24.
- API review (if needed) has been performed in order to graduate to beta
- Production Readiness Review (PRR) has been performed
- End-to-End (E2E) tests are expected

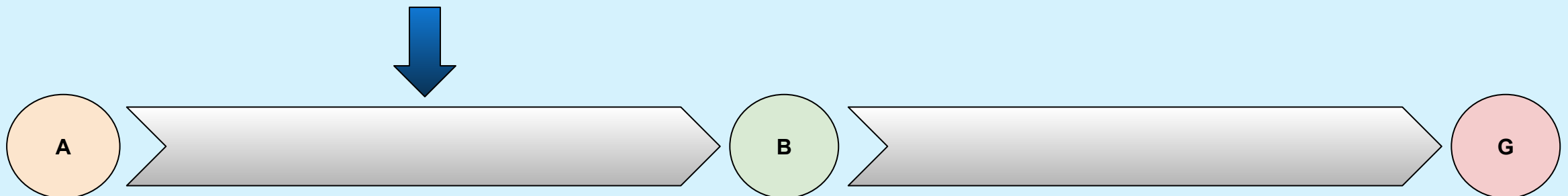


Beta stage KEP updates!

At each stage, the KEP introduced in alpha stage will require at least minimal changes to reflect the state of the new feature

Need to update the Production Readiness Review questionnaire, and pass the beta-stage PRR review.

Re-evaluate and incorporate alpha-stage feedback



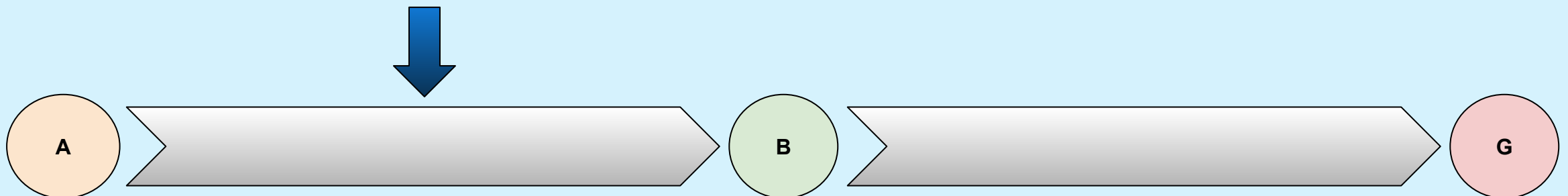
Feature gate enabled by default

Reaching beta stage means a significant milestone for the feature maturity

Beta APIs are however, NOT enabled by default

The road must be paved at this stage to meet the full requirements for GA-quality feature

This is what Production Readiness Reviews is for!



Production Readiness Review

Production Readiness Review (PRR) wants to ensure that Kubernetes features can be safely operated in production environments.

Goes side by side the KEP graduation from alpha to GA

Different set of reviewers - not from participating SIGs

Reviews are more and more thorough as the feature matures

Production Readiness Review

Production Readiness Review items to be addressed for Beta (and GA)

- Upgrade/Rollback
- Scalability
- Monitorability
- Dependencies



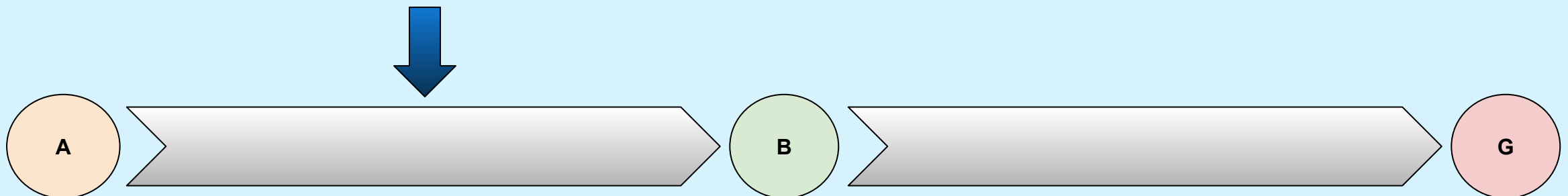
tip: review the PRR questions in the KEP template and plan ahead for these requirements. The sooner the better! PRR is required for **all** the stages of a feature

How did they do it?

Example feature 1: “[cpu manager extension to reject non SMT-aligned workload](#)”

Sig-arch conversation about the new 2 FGs on [KEP](#) ([PR](#), [discussion on mailing list](#))

E2E tests already added in alpha stage!

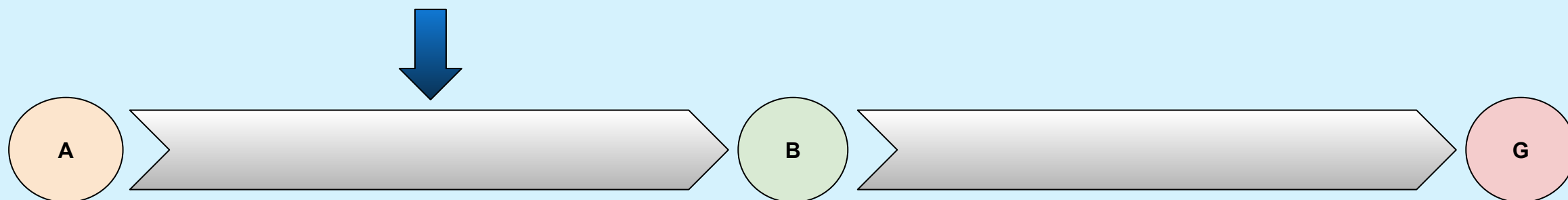


How did they do it?

Example feature 2: “[podresources endpoint to return allocatable resources](#)”

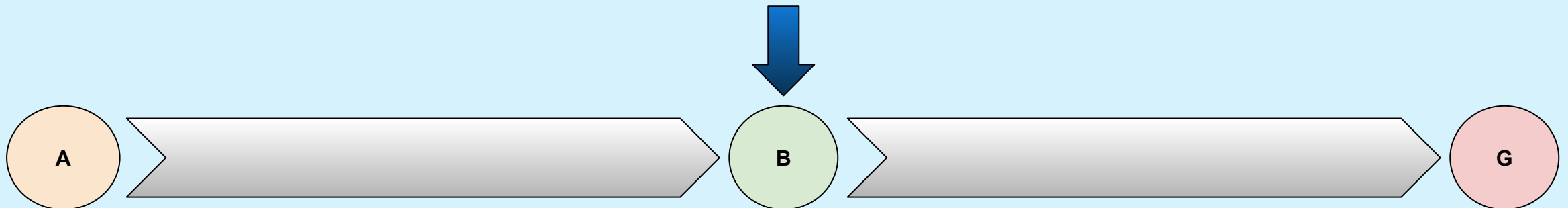
Key items: gather feedback (mostly fixes)

Key items: e2e coverage added earlier in the cycle

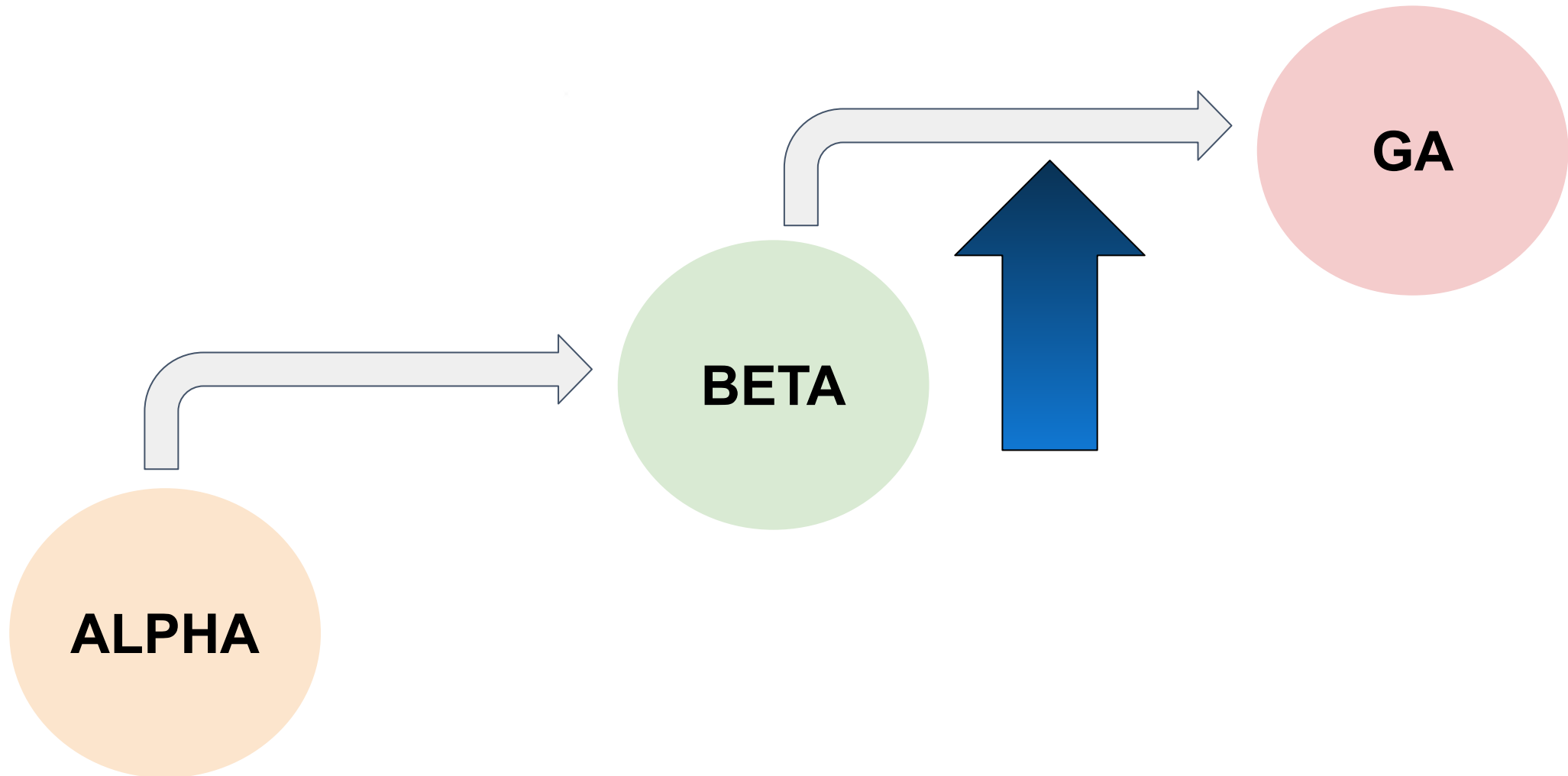


Beta Stage

Celebrate! The feature is Beta now! 🎉

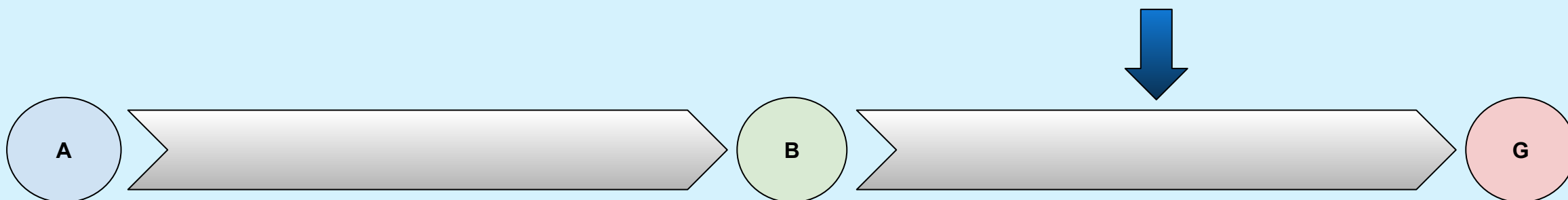


What happens in the post-beta stage?



The feature is expected to be stable

- [Audience](#): all users
- No impacts on cluster reliability
- Observable/monitorable
- Upgrade compatibility - handle version skews
- Deprecation path for removal
- End to End (E2E) tests

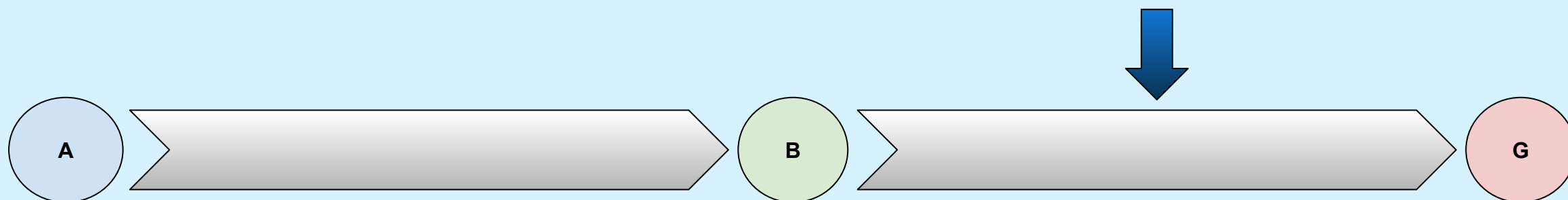


GA stage KEP updates!

Similar to beta and alpha stages, the KEP introduced will require at least minimal changes to reflect the state of the new feature.

Update again the Production Readiness Review questionnaire!

Incorporate feedback from beta stage, adjusting as needed



End-to-End (E2E) tests

End-to-End (E2E) tests: the upper layer of the testing pyramid

Verify user scenarios against a full-fledged system (cluster) by checking all the components of the system interact to produce the desired outcome

A special case: node E2E tests: the system under test is the node-local stack (kubelet, container runtime...)

E2E tests are useful, costly resource-wise and complex

E2E tests expectations

E2E test coverage is expected for Beta and GA features

E2E tests need to be reliable, avoid intermittent failures (flakes)



Tips:

- Add E2E tests as early as possible. Good investment: e2e tests verify behavior, not implementation!
- Periodically check the CI signal to minimize/reduce the flakes - CI can be a harsh environment!

How did they do it?

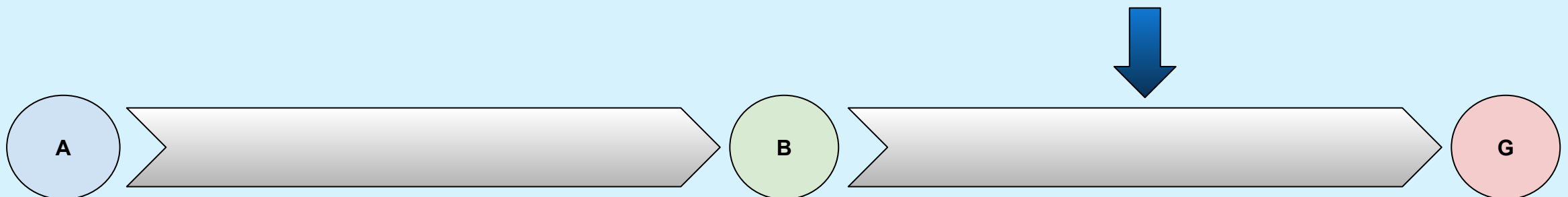
Example features:

- [CPU Manager GA graduation](#)
- [Device Manager GA graduation](#)
- [Topology manager GA graduation](#)

Key point: [avoid permabetas](#)

Key point: adopting KEPs

Key point: process (PRR) changed in between



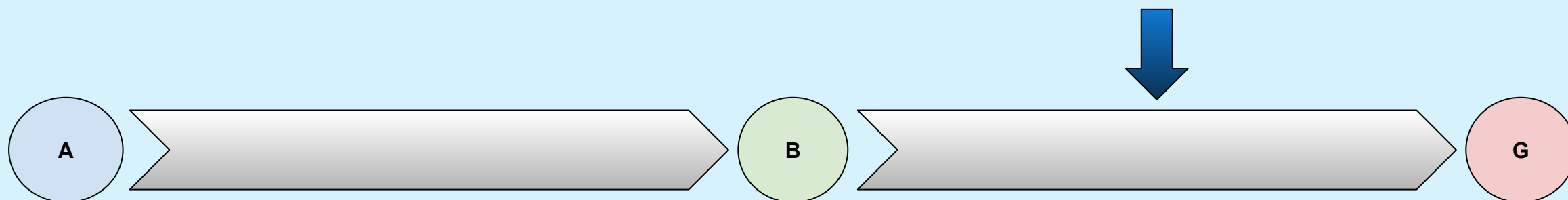
How did they do it?

Example feature: podresources GA (currently WIP)

Key point: adopting KEPs

Key items: gaps for GA graduation (windows support, rate limiting)

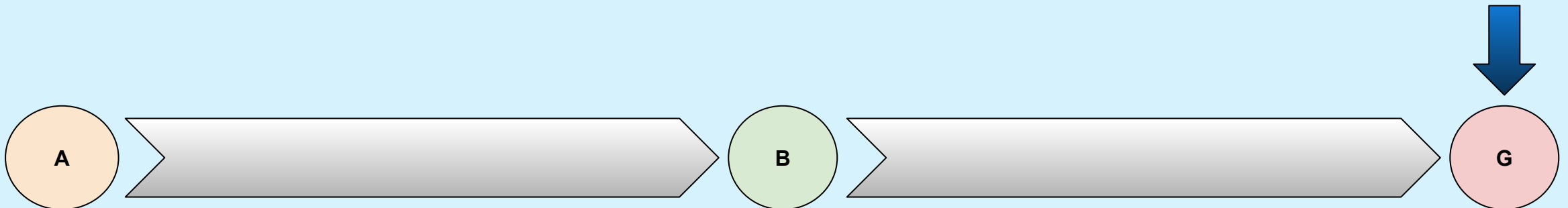
Key items: PRR review highlights



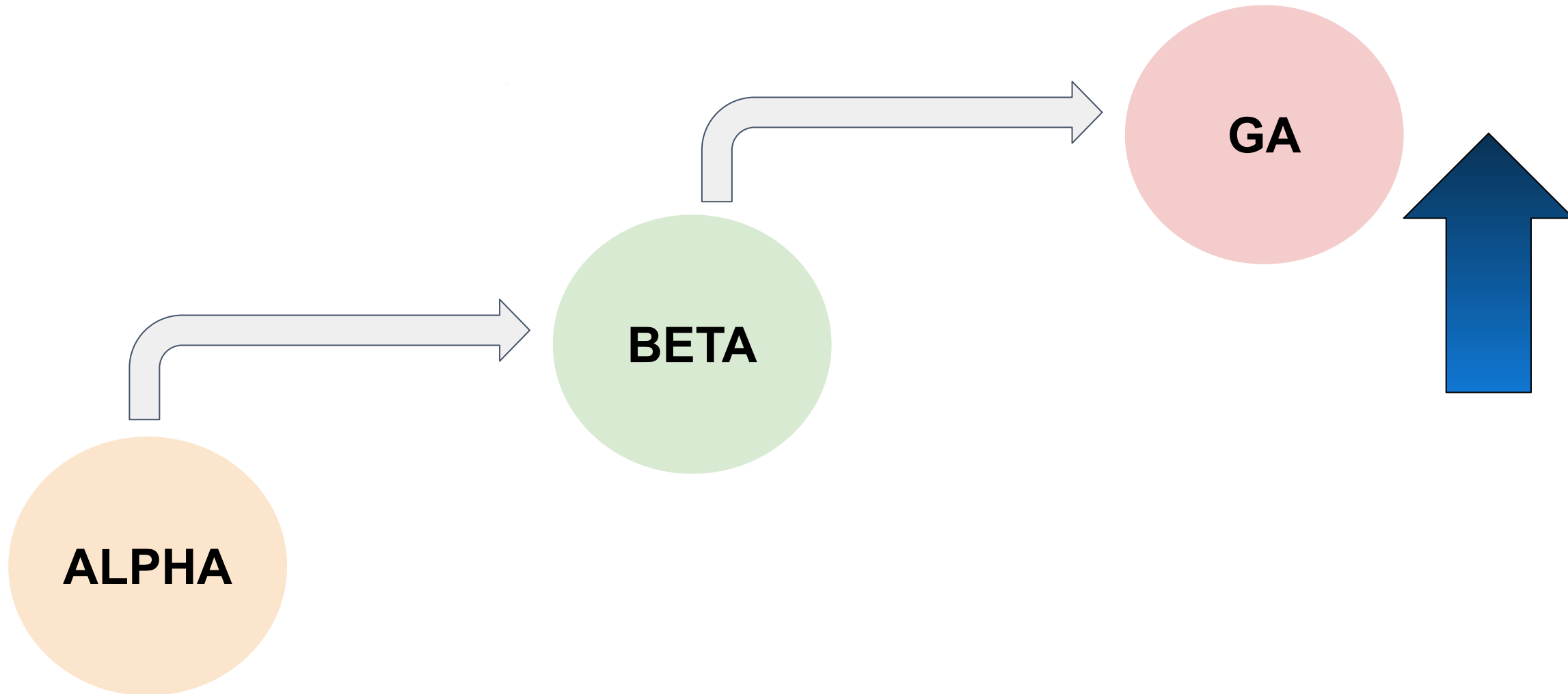
Feature is GA now!!

Mark the KEP as implemented

Celebrate! Well done! 🎉



What happens after GA stage?



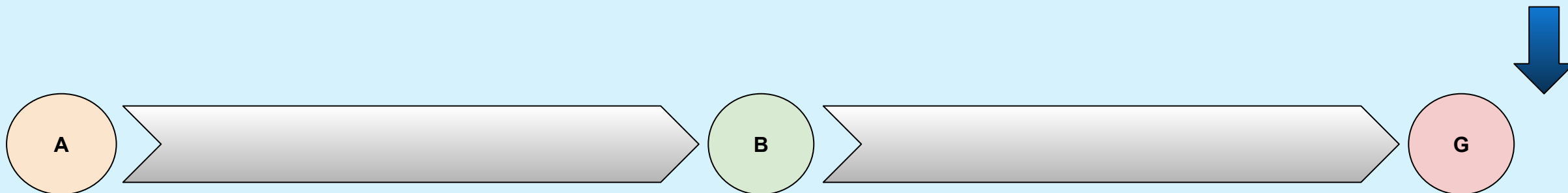
What happens after GA stage?

The feature is now exposed to the largest possible audience

Better to monitor issues for a couple cycles

End of the story?

1. Once a feature has become "implemented", major changes should get new KEPs.
2. A new cycle starts



Wrapping up

- **Communication:** being in touch with the community and be aware of the process and timelines will enable to anticipate issues as much as possible, and plan effectively
- **Persistence:** Kubernetes is run by a large distributed community spread all across the globe. Coordination of such large projects is challenging because logistical issues.

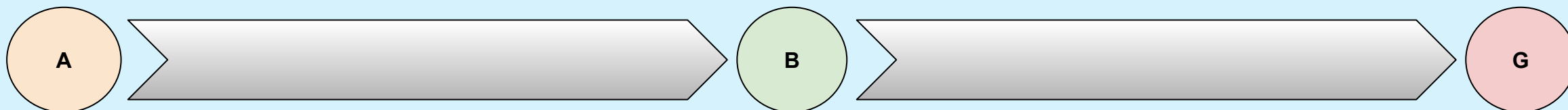
Parting thought - Take initiative

Identify areas of your interest and take **initiative** to help out in the community.

There are plenty of opportunities and ways to help out:

triage issues, work on bug fixes, propose new features, help resolve CI flakes etc etc...

New contribution in any shape or form are always welcome!





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