📝 Notes

Jun 5, 2025

CUE Community Update – Notes, automatically generated

Summary

Daniel Martí announced the release of Reserve 13 and the upcoming backporting release, while Marcel van Lohuizen detailed updates and future plans for the Evaluator, emphasizing performance improvements and the eventual removal of Eval V2; they also discussed the closeness algorithm and potential simplification of aliases. Paul Jolly and Marcel van Lohuizen initiated discussions on deployment patterns, defaults, and configuration packages, urging user feedback, and Roger Peppé presented the central registry's overview, publishing capabilities, and future enhancements, with Paul Jolly highlighting its open-source accessibility. Matthew Sackman provided an update on the Q Language Server Protocol's progress, and Paul Jolly and Jonathan Matthews highlighted recent documentation updates, while Dominik Mayer encouraged community feedback and contributions; finally, Steven Harris raised the topic of air-gapped module usage, with Roger Peppé and Paul Jolly outlining plans for `q mod download` and `q mod vendor` based on the OCI model.

Details

- **Release Updates** Daniel Martí announced that Reserve 13 has been released and a backporting release with bug fixes is expected soon, with 04 alphas to follow. Daniel reminded users to report issues with older versions, especially regarding the switch to eval by default, which can be turned off if needed (00:01:00).
- **Evaluator V3** Marcel van Lohuizen reported that V13 includes Eval V3 as the default, noting that performance improvements were deemed sufficient due to

the new closeness algorithm, though further gains are expected with memory management for this algorithm (00:02:02). Marcel mentioned ongoing work on performance issues with the closeness algorithm and the presence of some remaining Eval V2 and performance issues, with a proposal to not turn validators into ground values planned for evolving (00:03:05).

- **Evaluator Future Plans** Marcel van Lohuizen outlined future evaluator plans, including expert-specific disjunct elimination and addressing issues with unary equals, aiming for better error messages and considering changes to defaults and aliases (00:05:15). Marcel highlighted the need to fix bug 392 and plans to improve performance through memory management, fine-grained structure sharing, closeness algorithm enhancements, and utilizing discriminator fields, reiterating the plan to remove Eval V2 in V015 (00:06:12).
- **Closeness Algorithm and Aliases/Defaults** Marcel van Lohuizen discussed the need for disambiguation on export due to the new closeness philosophy and potential simplification of the closeness concept (00:08:41). Marcel also proposed replacing the seven different types of aliases with a single, simpler approach (00:09:48).
- **Importance of Unity** Marcel van Lohuizen emphasized the importance of using Unity for testing and feedback, as it helps in identifying performance improvements and regressions, leading to quicker fixes (00:09:48). Paul Jolly echoed this, thanking those who have added projects to Unity, noting its significant contribution to accelerating changes and understanding the impact of various updates (00:10:55).
- **Deployment Patterns and Defaults Discussion** Paul Jolly initiated a discussion on deployment patterns and defaults at the language level of Q, highlighting the complexity of managing multiple configurations, schema definitions, team coordination, and config sources (00:10:55). Paul posed questions about how to declare and structure complex environments, represent team collaboration, the role of default values, and reasoning about constraints, calling for user feedback and experiences in this area (00:13:12).
- **Call to Action on Deployment Patterns** Paul Jolly urged users to share their experiences with deployment patterns in Q, including what works well, what is challenging, and how Q can better support these scenarios, inviting them to join the #deployment-patterns channel on Slack or engage with the GitHub repo (00:15:28).

- **Experimental XML Encoding** Daniel Martí announced that the experimental XML encoding, previously available in an alpha release, is now included in the stable 013 release. Daniel clarified that this is one way to handle XML data in Q but is not the default encoding due to the complexities of mapping XML to JSON or Q (00:17:34).
- **Central Registry Overview and Publishing** Roger Peppé provided an overview of the central registry as a service for hosting schemas, noting its domain name-based namespace and current GitHub-based authorization (00:18:36). Roger announced that users can now publish their own modules under their own namespace by serving a configuration file from their domain that links to a GitHub repository for authorization (00:19:51).
- **Central Registry Future Enhancements** Roger Peppé outlined future improvements for the central registry, including delegation between domains, finer-grained access control, more documentation, the ability to import CRDs into Q, better documentation rendering, search and discovery features, GitHub Actions integration, and more authentication options (00:22:02). Roger also mentioned the availability of anonymous access to the central registry with rate limiting (00:24:20).
- **Open Source Projects and Central Registry Auth** Paul Jolly elaborated on the authentication aspect of the central registry, highlighting that open-source projects can leverage the public read-only access and that migrating to authenticated access is designed to be trivial using existing OCI mechanisms (00:25:36).
- **Q Language Server Protocol (LSP) Progress** Matthew Sackman provided an update on the Q LSP, which aims to provide features like automatic formatting, jump to definition, error diagnostics, and hover annotations in editors like VS Code, Emacs, and Neovim. Matthew explained that the LSP is being built by adapting the Go LSP implementation, which provides a solid framework but requires significant learning and tweaking (00:28:02).
- **Current State of Q LSP** Matthew Sackman stated that while the Q LSP can currently load modules and understand relationships between them, it does not yet provide any useful feedback to the editor (00:29:14). Matthew mentioned ongoing work to efficiently reload changed files and integrate them into existing packages, which is challenging due to the batch-oriented design of the current Q loading mechanism (00:30:29).

- **Q Loading Mechanism Refactoring** Paul Jolly emphasized that Matthew Sackman's work on the LSP's loading mechanism will also help address the overloaded current Q loading mechanism, potentially allowing for config options to disable Kubernetes-style loading and support different ways of loading Q code (00:31:44).
- **Docs and Content Updates** Paul Jolly and Jonathan Matthews presented updates to documentation, including a guide for upgrading to Eval V3, a TOML guide, and an analysis of Go API dependency changes by Roger Peppé (00:32:54). They highlighted the Eval V2 to V3 migration guide and several how-to guides on qlang.org, covering topics like embedding files and generating Go types from Q (00:35:18).
- **Q.dev Content** On q.dev, Roger Peppé published Kyverno guides that illustrate the journey of using Q for policy management with curated modules from the central registry (00:36:21).
- **Community Updates** Dominik Mayer reiterated the value of user feedback on use cases and challenges, noting that the authentication change in the central registry was a result of such input. Dominik encouraged users to add their logos to qlang.org to showcase Q adoption and help answer the "Who's using Q?" question (00:38:12).
- **Configuration Packages Discussion** Cedric Charly raised the idea of "configuration packages" for managing application configurations and their versions, suggesting leveraging the existing OCI-based module system with a potentially different cadence (00:40:12). Cedric also discussed the potential synergies between AI tools and Q for configuration generation and management (00:42:40).
- **AI and Q Synergy** Paul Jolly acknowledged the increasing use of AI LLMs and suggested that AI could generate Q configurations that can then be validated, creating a cycle based on facts (00:43:45).
- **Deployment Patterns Follow-up** Paul Jolly agreed that Cedric Charly's idea of configuration packages fits within the deployment patterns discussion and encouraged further input on different configuration sources and lifecycle management (00:46:03). Marcel van Lohuizen noted that there are multiple ways to approach configuration delivery and suggested an offline discussion (00:48:06).

- **Open Source Tooling Gap** Marcel van Lohuizen and Paul Jolly acknowledged a potential gap in open-source tooling for configuration packages, with Paul emphasizing that solutions might involve language features, tooling, and running services (00:48:06).
- **Prior Art: Flux Project** Steven Harris mentioned the Flux project's use of OCI artifacts for GitOps as prior art relevant to Cedric Charly's configuration packages idea (00:49:52).
- **Q's Role in Configuration Management** Paul Jolly highlighted Q's fundamental support for incomplete configurations supplemented in a principled way and language-level coordination between teams, especially through defaults and incomplete values (00:51:04).
- **Learning from Existing Systems** Paul Jolly emphasized the importance of learning from systems like Flux and others regarding what works well and areas for improvement in configuration management, particularly concerning coordination and conflict resolution between teams (00:52:24).
- **Air-Gapped Use of Modules** Steven Harris raised renewed interest in air-gapped use of Q modules, similar to vendoring, and highlighted the on-demand fetching of modules by the Q tool. Steven expressed interest in the proposed `q mod download` and other sub-plans to support this use case in environments like Basil where network access might not be available (00:53:29).
- **Q Mod Download and Vendor** Roger Peppé mentioned that `q mod download` and `q mod vendor` are on the roadmap, with the main task being to finalize the design of the command-line interface (00:54:38). Paul Jolly encouraged Steven Harris to add specific requirements for air-gapped module usage to the relevant issues for `q mod download` or `q mod vendor`, noting that other projects also need `q mod vendor` (00:55:33).
- **OCI Model for Air-Gap Scenarios** Paul Jolly reiterated that the OCI model is key to supporting air-gapped scenarios, allowing for mirroring of artifacts and local caches. Paul clarified the subtle differences between `q mod download` and `q mod vendor` and encouraged Steven to provide input on the desired functionality and user experience (00:56:37).

Suggested next steps

- [ ] Steven Harris will add specific requirements for air-gapped module usage with Basil to the Q mod download and/or Q mod vendor issue(s) on GitHub.

📖 Transcript

Jun 5, 2025

CUE Community Update - Transcript, automatically generated

00:00:00

Paul Jolly: video and audio now. And so we're off. Um, welcome everyone to the latest Q community update. Um, we're going to go through our usual sort of agenda here talking about release updates and the evaluator which is closely linked to the recent releases. Uh, a section about deployment patterns and defaults with a call to action there. Um, a regular slot of talking about the encodings that Q supports and changes there. Modules in the central registry. um updates on the LSP um as well as updates to docs and content on qlang.org and q.dev and then just sort of finishing up with a brief community updates and any Q&A at the end. Uh Daniel, I'll hand straight over to you.
Daniel Martí: Yep. So, in terms of releases, the last time we spoke, we were doing pre-releases for Reserve 13. That release is now out. It's been out for a couple of weeks now. Uh, we'll talk about the evaluator changes shortly, so I'm not going to go into those. And we also covered many of these changes um in the last community call.

00:01:00

Daniel Martí: So, I'm also not going to go into any of the details, but the change log is huge. So, I would suggest that you go look at it and uh think about updating if you haven't yet. And the idea is we will most likely do a backporting release in the next week or so because we we have had some bug fixes already. And then we're going to get the ball ball rolling for 04 uh alphas soon as well. And uh a reminder, if anybody's stuck on old versions, please let me know why. Um, I suspect that some of you may run into evaluator issues with the switch to eval by default. If that is the case, you can still turn it off, but please also let us know, either me directly or raising a bug report on GitHub, whichever way you prefer. But please let us know if you're blocked on an older release.
Paul Jolly: myself.
Roger Peppé: You're muted.
Paul Jolly: Oh,

Marcel van Lohuizen: Yes, muted evaluator. So, as Daniel already said, V13 has been released which uh includes making eval v3 the default.

00:02:02

Marcel van Lohuizen: So, um as we promised in later community calls, we said that uh we would need to do some performance improvements in order to release it. Uh in the end, we did not do it because the performance improvements for most most people was uh enough. uh this is mostly due to the new uh closeness algorithm. Um but that means that there's still a lot to gain. So whereas with the old um evaluator basically there was memory management um but not for the for the closeness algorithm. So that was relatively still slow but but so you know with the memory management this was was quite large. So now with a new evaluator um the the memory management for the normal evaluator blew up but now the closeness algorithm is much closer. So if we then also do memory management for the evaluator you can expect to have a significant performance improvement. So that's what we're still looking at. We also have some performance issues with the new closeness algorithm uh which I'm working on uh right now.

00:03:05

Marcel van Lohuizen: So it might become um even faster. It's affecting only only some people. Um, so there's many bug fixes. We're still uncovering more evolv bugs as finalizing the some other things. Um, um, but yeah, overall the number of bucks in evolv 3 is considerably lower than evolving. Um, so far we haven't we had a few uh reports of regressions, but uh, remarkably few actually. So thanks everybody for filing bugs again. Uh that's always very helpful and trying out a new evaluator. Um then yeah since last time we're still have one issue left which you haven't addressed but that didn't change. Um there is now only one high priority semantics f*** open which I mostly have fixed but it uncovers some other problems. These are the eval 2 problems that I was talking about. Um and some older issues that are still available uh still there in both. uh then we have two performance issues open which we as soon as that buck is out we we aim to fix.

00:04:12

Marcel van Lohuizen: So one of which is the memory management the other one is closeness buck and that still leaves us open the fine grained u uh um I call structure sharing as a possible uh optimization which also has not been done yet. Um so then towards uh 1.1 uh 0 so like like you know final version of the language a little bit longer term. Um but the next step uh I made a proposal a little while back to um to not turn validators into ground values. This has a lot of good properties to not do that. Uh it didn't affect Unity at all. I I ran the test but we it's been ready for a while now but we didn't want to co-mingle this with Eval V3 release. So this is plans for evolving but this is still something that needs to be uh done or submitted. Um so another thing that we want to do soon is um so ex um so now that we have released evolving 2 also um I think the title might be this is in the wrong slide but I'll mention it anyway.

00:05:15

Marcel van Lohuizen: It's the expert specific disjunct elimination. So with the new closeness algorithm closeness is a little bit less predictable sorry and with the export um so basically what we now want to do is if you export a value and you have for example unsatisfied required field or if if you have different um optional fields that that should be considered one value right so eliminating disjunctions in those cases so that's something that needs to be done another thing that we want to do is the unary equals equals But the first in column not one. Yeah. So right now if you if you have this um these two validators. So greater equals to one and less than equals to one. It simplifies it to one. But one is a concrete value and the validators are not. Uh and this is confusing to some people. We cannot always guarantee it. So it provides for inconsistencies. So it's it's it's not great.

00:06:12

Marcel van Lohuizen: Um so error messages and friends. um this is something we want to do for you know like like sooner rather than later. So another thing is uh Paul will talk about it later. Defaults and aliases we want to change a bit how that work. Uh the use of discriminator fields and um a try proposal which I've written and hope to to publish uh soon. Um which is also some some quite uh handy language feature. Um so for the next steps for the evaluator itself. taking it step back. It's there's one this this this one remaining bug 392 uh to field not allowed aggression. Uh we really need to fix that. Um um so that's the one Unity uh buck that's remaining. So it's mostly done. Um and also what we've noticed um is that this might break more evaluing it. But these are all bugs in EVA v2. So like really quite serious bugs in Evolve V2 that we can't really work around.

00:07:15

Marcel van Lohuizen: Um but the other thing performance-wise as I said memory management fine grain structure sharing closeness algorithm but also discriminator fields. Rush has done some work on uh detecting discriminator fields and we can use that to uh pre-eliminate disjunctions. Um so I'll add this warning on the slide multiple times. It's not an accident. So we plan to remove EVOL v2 in V015. So just as a heads up. So if you keep relying on evolve v2 at some point it will disappear. Um so towards uh 0.14 specifically. So before I mentioned next steps could in 013 um but one of them is what I mentioned earlier the the the manifestation of bounds. But the error messages we really want to do now. We also want to take uh so this is custom error messages um and some other uh reflection uh built-ins uh I think it was 843 issue so improved error messages um especially when we start removing zero or eval 2 it will be easier to do but even before that with this with the discriminator field support we can start looking at uh improving error messages um so Another thing that's really important is the disambiguation on export.

00:08:41

Marcel van Lohuizen: Um because this so with a new closeness the new philosophy of closeness that doesn't the way it works right now doesn't jive fully with it anymore. It already didn't in EV in the evol but but I think at least from a philosophical point it's uh it's important to accelerate this now. Um um and this will be a strict improvement I would I don't expect there to be incompatibilities. Um so the unary unary equal support uh might be a bit early for V014 but we we could consider it. So another thing the whole closeness algorithm was designed around a different concept of closeness and we're looking at just simplifying closeness altogether making it much more explicit to so to not assign spec specific meaning to the embedding position of references but having basically if you have a reference and it's added with a dot dot dot it basically means don't check closeness and without it's always checked regardless of whether it's an embedding position or not but that's a big transition phase so we'll tricky um to manage but it should be possible.

00:09:48

Marcel van Lohuizen: Then aliases and defaults um um so aliases another proposal you might expect. So in this detect there's now seven different types of aliases and that's they look all the same but they're all different and it's quite confusing to user. So we have an idea to replace this with one and we can uh do this which I think will make things a lot simpler. um slides. So, and then finally, I want to remind people to use Unity. We've heavily relied on this for Evolve V3. Um we picked Unity projects for uh finding performance improvements and we did those were typically the first ones we tackled. It's also easy for us to check if we had success and also with regressions uh you know before we submit something uh we'll will catch the problems with with uh in unity right some cases we choose to to not um how do you call it to not um to still put a regression in unity uh but you know this is this is quite rare we always make sure usually always make sure that unity keeps passing um and then even if you very quickly reported to us, right?

00:10:55

Marcel van Lohuizen: Like it's an out of sight, out of mind, right? So it can take like a another cycle before we get to fix whereas with Unity the feedback is very uh very direct. So a lot of benefits for you to to use it. Um so here's the link and I think that concludes my slides.
Paul Jolly: Thank you very much. Yeah, just to reiterate, huge thanks to those who have added project to Unity recently. It massively improved our ability to uh not only accelerate changes to eval bug fixes but also to um understand the impact of various changes whatever they might be and that is again just tightening that loop on us being able to make changes quicker and even proposals as well. So thank you. Um so now just talking about a bit about deployment patterns and defaults. Um and here uh what I'm talking about is defaults at the language level of Q. So default values as they're referred to. Um so um just a quick state of the nation on the kinds of scenarios that we're all dealing with on a daily basis.

00:12:00

Paul Jolly: Some combination of the following whether it be multiple configurations involving multiple environments, regions, whatever they might be. Schema definitions for our config inputs and outputs. coordination between teams, policy enforcement, whatever it might be, multiple config sources. This is what we're dealing with. And this is actually one of the benefits that we have from Q is actually introducing some sort of um description or declaration of how these things combine together and how they unify and what the inputs are and what the outputs are. Um but there are a number of still a number of open questions with all of that. Q at the language level is sort of at the base of all of this and then we have things like tooling and modules etc that builds on top of that and so what we're trying to establish is well what are the the right building blocks what are the right primitives for achieving all of these things because we're not going to solve everything at the language level some things will be solved in the language some other things will be solved in tooling other things like version control or coordination is solved by modules and so it not everything has its solution in the language And with this in mind, what we're really looking to do is just express a few questions here.

00:13:12

Paul Jolly: The number of open questions. How should these deployment patterns and not be restrictive in that term either, but how should we declare and structure these sorts of complex environments? How best to represent the collaboration between teams and all of this? Now comes sort of quite a language specific thing. What's the right role for default values in all of this? Default values at the language level. Are they good enough right now? Do we need to rethink or extend them in some way? Do we need to sort of think about a concept that's at sort of like some other slightly higher level when we're doing answering some of these questions? I we don't have to solve everything at a language level. We can as I said solve different problems at different levels here. How do we reason about constraints between different components in any such system? So there are some there are a large number of these that sort of have either direct answers or hints at answers in Q the language itself or indeed Q the language plus some tooling or modules but it's it's all about now sort of trying to tie these things together particularly when there are a number of users out there who have very slash extremely large Q configurations how actually do we start to sort of establish embody and

00:14:27

Paul Jolly: fully support best practices in all of these areas is here. So this is really a call to action. What we're really asking for here is especially for those people who have some experience of Q and I don't want to limit it to at scale. It's just some experience with Q. It could be very little experience with Q relatively speaking as in you tried defaults you tried and they just didn't work for you. That's absolutely fine. or to the other extreme where you have 40 teams and you know x number of people per team they're all working with Q and you're having to sort of coordinate between them what is your experience in this space what patterns are working well for you what as we say here is messy manual or brittle where there's friction let us know is Q getting in your way when really it shouldn't be it should be there as a tool to sort of support all of this stuff going forward and then sort of slightly more abstractly, how are you managing the sort of the the very real human aspect to this coordination between teams?

00:15:28

Paul Jolly: What who can see what etc. How do you want to see defaults slash layering whatever layering might mean to you? Do you use a different term? Just all of this is is as you can see intrinsically linked to changes within the language when it comes to things like defaults but also potentially other layers that build very closely on top of this. So this is a call to action and a request. We have already started back in February I think it was discussion in this space here and a big thank you to um Joshua Gilman, Tom Vanintha um uh Presanth and a number of others who joined that call to really just sort of kickstart this conversation. It was a very loose open discussion at the time. But now that we've gotten to a point where eval v3 is getting well it's already on by default in the latest version right. So we're sort of creating space as Marcel suggested to be able to work on these language changes but also to tackle these slightly larger pieces that build on top of what are just purely language constructs to actually solve real world problems bluntly with more than just language changes.

00:16:35

Paul Jolly: So this is what we're trying to tackle. So please join us in the deployment patterns uh channel on Slack. There's also a repo which is it's not really being used at the moment but it's it's intended to be very open for issues PRs whatever of people to just share ideas on how they think about things what they're doing in this space and I'm looking to help sort of kickstart or restart engagement with that but feel free to just message on Slack as well or Discord raise an issue in the main Q repo or if there's anything sort of slightly non-public that you want to discuss because often in these situations with when it comes to configuration that is the case please feel free to just get in touch directly as well. Um, so that's a call to action on deployment patterns and defaults and sort of anything related. So please get in touch on that. Um, encodings Daniel, can I hand back to you on that one?
Daniel Martí: Yep. So this will be another brief recap because the last time we did a community call, there was a new experimental XML encoding that was available in an alpha release.

00:17:34

Daniel Martí: So that has shipped now in 013 which is a stable release. The encoding is still experimental but you can play with it. Um and there's an example right there and you can follow the discussion for more details. Um we plan to transition this encoding to be stable at some point but for now it's still experimental. And just to reiterate that this is one XML encoding for Q, but this is not the default encoding. That's why it's XML plus Koala. Um and and as a very brief recap, that's because XML doesn't map to JSON or Q cleanly, right? Because of attributes and so on and name spaces. Um so this is
Paul Jolly: Awesome.
Daniel Martí: one way to deal with XML data in Q experimental for now.
Paul Jolly: Thank you very much. I think we're now on where are we? Modules of the central registry. Roj.
Roger Peppé: Uh yes. So um so I'll go over some of these points will be familiar but just for those who haven't uh haven't been to one of these meetings before um a reminder of what the central registry is.

00:18:36

Roger Peppé: So it's a central uh service on the internet um for hosting some uh schemas. they not necessarily well known and not necessarily well-known services but but particularly it's well-known place and many well-known schemas and well-known services uh schemas for well-known services will be there um it's uh the name space is like um Q modules are it's domain name based but those domain names uh uh signify where it comes from not necessarily what what what the source is of the particular data there um and at the moment Uh well actually that point is now out of date because we are we do allow as I will point out in a sub slide non-github.com rooted modules for now I'll go into that in a little bit but all the authorization and authentication is GitHub based still but that will change um and and uh yeah it's all servo based similar to go uh go version module versioning um and you can support publishing. Uh you can you can publish your own modules and you can derive some modules from elsewhere and publish those too if you like.

00:19:51

Roger Peppé: Um now oh I can actually the arrow thing does work good. Um now uh yeah big news if you want to publish your own modules under your own namespace you can actually do that right now. We are doing that. Um uh the way to do that is if you're in control if you're serving some content from your particular domain um you can serve something up inside the well-known uh well-known URL and I'll go to a little bit of detail on that in a moment. Um and that will map your domain to a GitHub repository that is then used for authorization for all authorization decisions uh on that on that domain on that module domain. Um, and as I said, we're using that in Q.dev. So, here's an example. You can that's that's all you need. You just need to make that curl command work for your domain. So, in this case, we're doing https fetch of q.dev and then that that that well that um particular phraseq
Paul Jolly: Heat. Heat.

00:20:54

Roger Peppé: central registry.json and you need to just serve up something that says we're the only type we support is GitHub. So that type GitHub always has to be there and then you give the name of your GitHub repository which is that's what's going to be used for authorization decisions whe when when pushing to or pulling from that particular domain. Um and for now there is only one authorization one repeat repo for anything in that domain. That will probably change in the future but for now we're starting simple. Um, and then there's a hint to the cache uh to say how long how long you want that to that to be. So we we we will probably um leave we won't we won't pull it that often basically. Um uh so here 24 hours um and uh in the future we will probably support uh text record support um rather than rather than HTTP support similar kind to the kind of way that um DNS uh like SSL certificates work uh for with um with let's encrypt or blue sky works.

00:22:02

Roger Peppé: So I think that's that's a fairly standard thing to do. We'll probably do that. Um, we'd like to provide the ability for to delegate from one domain to another. Um, there's uh and and more finer grade access control so you can have maybe several repositories uh within within your domain. Um, but uh but for now we've kept it simple. Uh yeah, please try it out. See what you think. Um and uh and and we've got a discussion uh that will be linked through there. So you can leave any feedback um for us, please do. Um another thing that we've done is published one example CRD um uh for to to uh to the central registry. We want to publish lots more, but we're working on some improvements. Um so there's an example there. We've just published published some Kyivero CRDs. Um and uh as of uh earlier today we've now that those actually fill in uh CDs actually fill in the API version of kind fields which uh actually is is pretty useful.

00:23:09

Roger Peppé: Um uh so so so you you can't get those wrong and your error messages will uh will uh if you if you if you think if you you're providing a uh if you're checking a CRD that has the wrong API version field fitted um set then it will tell you directly about that. again feedback welcome and um we'd like to know what other um what other CDs you might like to see in the cur cured modules and I've added I've created a discussion for people to send that kind of feedback please do um and as far as where we're going next um we want lots more documentation about using the curated schemas and using the set general um generally uh we uh we want to make um provide the ability to to import Cids into Q without because at the moment it's mostly custom inside inside the central registry code. Um and we want to improve reg uh in general rendering of documentation for for modules that are in the central registry. That would be that's that's doing a lot of work on that and search and discovery.

00:24:20

Roger Peppé: Um, and we want the ability to have have your GitHub actions able to um just get central registry access automatically. Um, and and more authentication options too, not necessarily just GitHub. Um, uh, and if you want uh other schemas in uh in part as part of curated modules, please let us know. Uh one thing that I haven't that I haven't mentioned I don't know why I think I thought I did is that we we do have another authentication option now uh we actually have the option to of no authentication at all. So we actually do support uh anonymous access to the central registry. It's it's fairly rate limited but it works. Um so so if you if you're if you want to do stuff in GitHub reactions that involves pulling modules public modules from the central registry you can just do that now. There's no need to send a token. There's no you can just do that. Um if you want uh the the rate limits to go away then of course um you'll need to you'll need to uh authenticate with a token but uh that should be useful and that means that for example open open source projects don't necessarily need um to people to authenticate when pulling them.

00:25:36

Roger Peppé: So I think that's a that's a big improvement. Um and I think that's the end of my update.
Paul Jolly: it is just to briefly add on that on the piece on the orth side of things. Um yes, exactly. So the open source projects um there's at least two projects that are relying on the central registry's l lack of requiring orth now i.e. there being public readonly access in their project. So they then just can start using the central registry um straight off the bat to their users without needing to sign up to any sort of service which is a good thing for the open source projects perspective but it we're going to actually add some docs around this as well. If those projects and their users actually do need orth, um the migration from using no orth to actually using any form of orth is trivial for either the Q

command or anything that wraps the Go API. It's this was a a key part of the design of the modules proposal being OCI based that people can and should be able to leverage existing OCI mechanisms OCI or mechanisms without with almost zero effort and so the central registry being just a special case of an OCI repository that we have GitHub based orth for now and as we said we'll do other orth options but if someone say for example in whatever enterprise already has an

00:27:02

Paul Jolly: orth setup, those open-source tools that can just rely on public read only access to the central registry can just trivially be pointed at say for example an internal OCI registry and just work by setting like a config variable. So this transition from no orthor required to oh I want to use this specific OCI server is designed to be trivial when it comes to the module side of things and that was a key part of the modules proposal. So that really sort of supports those open source projects both ways. There's almost zerocost adoption of trying out the central registry but those same open source projects their users can flip to point at an internal registry if that's what they actually need to do. So, we'll just be providing some more um docs around that space. Um there was something else that I was going to mention off the back of Roger's update, but I've totally forgotten it now. So, um we can come to that later if I if I remember it. Otherwise, I hand over to Matthew to talk about uh QLSP

00:28:02

Matthew Sackman: Cool. Thank you. Um, so yes, so, uh, just in case you don't know, the LSP is the language server protocol and it's how your editor, well, modern editors, uh, get autocomp completion and jump to definition and those sorts of functionalities. Um, so I'm working on the Q LSP. Um so the sort of initial minimal set of features that t are being targeted are things like automatic formatting on save um jump definition uh error diagnostics um annotations of when when you're hovering over stuff in VS Code that sort of thing. Um and we'll be testing against VS Code, Emacs, Neovim, and possibly one or two other editors as well. Um now we're building this by taking the um the go LSP um implementation um and adapting that and that has pros and cons. Um it gets us an awful lot of machinery and sort of well tested and well um un well well well well used um framework for for relatively low cost. Um but you know obviously there's actually quite a lot of code there to um learn and understand and tweak.

00:29:14

Matthew Sackman: Um, so I've been, you know, slowly trying to make the the two worlds meet together in a in a sensible way. Um, so currently you can use Q LSP. Um, though really all it does is when you um open a Q file, um, the LSP server gets told about it and it will load that module um, and that's it. So you get absolutely sort of nothing useful um yet. Um so don't use it. Um it understands relationships between modules and packages. Um so for example it builds up the dependency graph. It understands when um you know one one file in a package changes and therefore it has to invalidate not only every not only all the cache results within that particular package but also cache results in any other package which happens to be loaded that it that imports that original modified package transitively. Um so a bunch of that machinery is is done and seems to be working. um it can understand in some cases when some of this cache stuff needs to be invalidated.

00:30:29

Matthew Sackman: So in particular if you're editing a file and you change the package that this file is in um then it will spot that that's happened and it will um try and do some uh some of the recalculations. But really at the moment all it's doing is is just reloading the packages. It's not actually doing any sort of analysis and it's certainly not sending any useful information back to the editor at this point. Um, in the last couple of weeks I've been uh looking more at um in the LSP we need to be able to cope with oh a file has changed. We need to be able to very efficiently reload that file, recombine it into um existing packages which have already been loaded. um and we want to do the least amount of work possible because you know you're you're responding to keyboard key presses. Um whereas in the Q world um an awful lot of the code around uh loading packages and modules has been written much more sort of batchy um in design because it's mainly used from the command line where where you know you just want to load everything as quickly as possible and it doesn't really support the um oh this file has changed can you just re recmputee this sort of thing.

00:31:44

Matthew Sackman: Um, so yeah, there's there's quite a lot of low-level plumbing going on to try and make that work a little bit better. Um, so yes, that's that is the state of play right now. Um, hopefully by the by the time um the we we have a next community update call, there will be something uh more useful to show.
Paul Jolly: That's kind. Um, just the the last point there I think hints as to again where we've been running up against sort of another challenge that is open as well as to what it means to actually load Q code and that right now we only have this one loading mechanism that is in some situations hopelessly overloaded. if you'll excuse the pun. Um, and that actually we need to piece that apart, provide a config option to not do the Kubernetes style ancestor directory loading, but also then extend to support and this has been a long-standing request different ways of loading Q code. So having a single package essentially split across multiple directories potentially. And so basically the work that well not basically that that definitely um underplays it.

00:32:54

Paul Jolly: The work that Matthew's doing here to relook at the loading mechanism also provides us a means of tackling two birds with one stone because quite frankly the LSP will crumble under the weight of loading too much Q if that's actually not what the user intends and it won't even work for projects where they don't load according to a standard pattern. So we need to support these standard patterns. So at least this refactoring also provides us with a clear opportunity to branch and actually introduce different loading mechanisms as well. So that's a key part of the work. Um super thanks Matthew. Um moving briefly onto the docs and content side of things and a big thank you to uh Jonathan Matthews for helping to not only make a number of changes in this space but also for pulling together this um these slides here. Um we've talked about the upgrade to eval of uh 0.13. And so we have a guide that sort of touches on the key things to consider when upgrading to 013. Um we have a TOML guide that is again just looking to advertise the ways in which Q can work with different encodings.

00:33:56

Paul Jolly: TOML in this case and that similarity with YAML and JSON. Um, Rod has pulled together a first cut of a uh concept piece that looks at how when using the Go API, your dependencies in your Go project actually change. Uh, and this was um in response to some great feedback from Eddie Knight um and others who contribute to his project whereby they looked to add the evaluator to their project and they saw you a big diff when it came to the go. file. And unfortunately, when you're just seeing a diff at a file level, that doesn't actually tell you very much um about what it is that's going on from a dependency story. So, Roger's done some really nice and uh some nice analysis in there with some pretty graphs that help to explain and sort of piece apart pe or not piece apart, sort of piece out the the bits that are um test only dependencies or whatever they might be. And so that's a a really nice explainer that also then gives a visual idea of okay what is your actual for one of a better phrase security exposure when it comes to um adding Q as a dependency what code are you actually running um and Ro also hints at the bottom of that article it's something else that we thought about for some time is to how we can best sort of layer dependencies potentially shifting towards a

00:35:18

Paul Jolly: a dependency injection style model where you can truly have minimal dependencies when it comes to whatever you want to do with cube. But that's just a sort of a thought phase right now. That document will hopefully um give some actually hints as to what we're actually doing there. Uh and here is that guide from Eval V2 to V3. Uh so please go and give that a look especially if you are in the process of trying out is in combination with the release notes. It's designed to sort of com It's designed to be a partner to those release notes to sort of highlight the key arrays, especially where there might be semantic differences. Um, a number of different how-to guides on qang.org there. Um, one perhaps favorite that I have amongst that list there, not that I should necessarily have favorites. Um, embedding files is we're still making heavy use of embedding. Um, generating go types from Q definitions. That's something that we are heavily dog fooding ourselves. Um in in internal services that we've written um we used to go the other way round.

00:36:21

Paul Jolly: We used to start from Q excuse me start from go and go to Q. Too many goes in there but hopefully that made sense. Um uh but now actually we use Q as the source of truth and generate go which is a much cleaner approach because it allows us to maintain that logical single source of truth much better. Um, a number of folks have asked questions that basically I say, well, hey, we how do we get dependencies between one OCI server and another and there's a nice piece there about Qod mirror that explains again it sort of ties into this story of the OCI model behind uh Q's dependencies. So that's worth a look as well. Um, so yeah, I can't really say that I have favorites because I've listed three or four there. Um, on the QDV side of things, uh, as Raj said, we've got the Kyerno guides. He hinted that them being available soon. They are there already. So, please give them a try. These are similar to the guides that we have for the other curated modules that we've published.

00:37:19

Paul Jolly: Um, and that had a sort of each of them is trying to sort of introduce this concept of what the journey with Q is where you perhaps start with an existing in this case policy start which is YAML based. How do you actually validate that? How do you start to move more towards Q as a source of truth and what benefits you have from doing that along the way? And all this all these examples use um the curated modules that are published to the to the central registry. And there is that page with a nice little screenshot there. Um that's it on the doc side of things. Um Dominic, if you want to take everyone on the community update.
Dominik Mayer: Yes, a short community update. H can I move to the next slide?
Paul Jolly: I can for you. Apologies.
Dominik Mayer: Thank you
Paul Jolly: I
Dominik Mayer: very
Paul Jolly: will
Dominik Mayer: much.
Paul Jolly: add you.
Dominik Mayer: So in this period there was no conference so no merge updates but uh very similar to previous updates.

00:38:12

Dominik Mayer: I just want to reiterate how helpful it is to learn about your use cases, challenges and what you would like to have from Q. For example, the authentication change with the central registry. That was one learning we received by talking to people and we have a lot of conversations with users, potential users who learn a lot and that helps us a lot in the daily work. So if you have anything you want to discuss, just don't hesitate, let us know. And another thing is logos on qing.org. We've added a few logos over between the last community call and now it's extremely helpful. If you use Q and want to display your logo, please do so. It also helps us because one question we often get asked is, "Oh, who's using Q?" And it's always easier to just show this page and say, "Oh, there's a a long list of users. I know already the list is way longer, but it's nice to have something we can publicly share." So that would be appreciated. If there's anything just always please feel free to reach out.

00:39:15

Dominik Mayer: Thank you very much.
Paul Jolly: Yeah. and to say we're also looking to work on those uh case studies that actually provide the detail behind those use cases as well. U and for example with um uh apologies I there's one logo that is in the process of being added which I won't preempt
Dominik Mayer: Next
Paul Jolly: so
Dominik Mayer: community.
Paul Jolly: I'll skip that one and then
Dominik Mayer: Next.
Paul Jolly: we can talk about that on the next uh community update. I just caught myself before I said it out loud there. Um I think that's it. So if anybody has any questions at this point, we can um switch to those. Although thanks to those who have been asking questions along the way and thanks to Raj Marcel who look who have clearly been answering them along the way. But if there's anything that hasn't been answered, please fire away. Uh Cedric, please.
Cedric Charly: Hey, it's been a while. Um,
Paul Jolly: Indeed.
Cedric Charly: one thing I've
Marcel van Lohuizen: I

00:40:12

Cedric Charly: been thinking
Marcel van Lohuizen: said
Cedric Charly: about, hey, um, I know we've been working on the module systems. Great idea. Something we needed for a long time. Um I think one thing that there's kind of a gap in our current I guess in the current software tooling uh ecosystem is I guess you'd call it configuration packages you know where um we need a way to get these whatever arbitrary configurations that are fed into applications as well as um the different versions. It's not something like a lot of people right now I'd say the simple way is to use environment variables. Um that has some security implications that make it not ideal. So the recommended thing to do is to use a file system or like use a file that's loaded into the application at runtime and then um they usually mount that on a Kubernetes volume but then also like the Kubernetes secrets itself um is not necessarily actually secret. So it can be awkward. Um I know the closest thing that I can think of that actually works is how I'm saying is crossplane has a concept of configuration packages and it also uses those OCI container images as the kind of delivery mechanism and so this would be kind of something similar to the module like it's still using the OCI container as a format but um I guess it' be a different cadence because you know when you're publishing when you're public

00:41:40

Cedric Charly: sorry when you're publishing ing a module for public consumption that's kind of on a slower pace.
Roger Peppé: Heat. Heat.
Cedric Charly: You're trying to be careful the changes. This is something that's tightly tied to specific application and you'd be like maybe even deploy multiple configuration versions per application version depending because you might want to change that faster, right? Um yeah, and then dealing with all the secrets
Paul Jolly: Thank you.
Cedric Charly: and stuff. I think there's a there's a gap right now that could be useful for Q, but um we already done the work to do the uh module system and we could probably leverage that same infrastructure. It just be kind of a different modality if that makes

sense. So that might be an interesting thing to explore. Um I'd have to think about a little more on the implications. So that's one thing. Um, another thing is I've been playing around with a lot of the different AI tools. You know, it's a very rapidly changing environment. I'd say right now the best one is Cloud Code with the the Cloud 4 models.

00:42:40

Cedric Charly: It's actually pretty incredible. I've been using like I I got the max version of everything on all like the OpenI stuff and like that like that. So, it's a it's pretty incredible right now. has been thinking about like when you think about how we use configuration and what we're trying to do with Q is you know see we want these libraries we want well structured schemas for things and then there's like the last mile where the developer per application per whatever thing you want to configure um they do the work but I feel like AI is very good at this um just generation in general but also just like figuring out the structure of an application like you can now create tasks for it to hey look at this application and create the configuration that will wire stuff together. It's actually really good at that. And so, um, I still think there's still use for, of course, the schemas and stuff. Um, there might be some use cases where instead of configuring things in a separate language, it might be just faster to generate it inside the application code because you can rapidly just ask the AI to like change it on demand.

00:43:45

Cedric Charly: But there's still I think the need for again versioning of specific configurations and dynamically reloading it as well as secrets or um gluing between different components and complex systems you know data intensive applications that's not going to go away. So I think I think finding those synergies at where we can um leverage that to create demand because again like both these things configuration packages as well as making it ideal for AI use cases would I think drive a lot of adoption for Q which is why I'm bringing them. So that's just two things I'm
Paul Jolly: Thanks.
Cedric Charly: exploring right now.
Paul Jolly: That that's absolutely wonderful. Thank you. Um I'll I'll allow others to to to chip in if they like or indeed we could sort of certainly follow up afterwards. I'll tackle those in in reverse if you don't mind. Honestly, I think the the extent to which in places

um I personally but sort of I know others are using um AI LLM on a daily basis is just increasing that that's that's super clear as as like a supporting tool from my perspective in in in many ways.

00:44:57

Paul Jolly: um the way that I um so so I think there's like a huge space to explore there but the way that I conceptualize things with respect to Q and this space is that um yeah there's a number of things where none of these AI tools are going to be perfect but what's great is that if actually they're generating say some Q um and then I can confirm though that that generated queue is essentially facts and confirm that as a human and then use those facts as the basis for further stuff to happen. That's great because then you've got some sort of cycle going on there. Yes. Where there's generation going on, but it's based on facts. Facts that I as a human and guess what? Q validation can actually confirm to be true or at least valid according to the statement of those facts. Um I I was just going to say um I think Nick and others are doing a huge amount in this space. So I would definitely encourage folks to sort of if they're interested in doing so and willing to to share on things like Slack, Discord, set up other sessions, right?

00:46:03

Paul Jolly: Because I I I hate to think we're never going to fit in the remaining 12 minutes on this session. The huge amount that I know others are are spending on this space here. briefly to just talk about the first point that you mentioned um Cedric regarding the um I think you call them configuration
Cedric Charly: packages.
Paul Jolly: uh packages. Thank you. That was the phrase I was after. Honestly, that that's perfect u feedback and I think it fits squarely within that call to action that I had earlier on on the deployment patterns piece.
Cedric Charly: Right.
Paul Jolly: um that that configuration packages thing whether they are bundled as OCI or as I said you actually have some source that is even a database which is receiving you know many updates potentially per second. I I think you sort of kind of hit the nail on the head there is is that there could be a plethora of different sources that end up sort of feeding into some evaluation and that you know that just changing over time and

and so yeah please do if if you've got specific thoughts or ideas in that space let's let's pick up that conversation in what I would call what we're going to call the deployment pattern space because I think it's just useful to think about how in my head I would consider as such um a thing being like a different source for such a system just as like a GitHub repo would be a

00:47:25

Paul Jolly: different source just as like a an SQL database w with some sort of timebased um trigger or whatever would
Cedric Charly: managing
Paul Jolly: be as well
Cedric Charly: the
Paul Jolly: right
Cedric Charly: life cycle
Paul Jolly: it
Cedric Charly: for
Paul Jolly: yeah
Cedric Charly: sure.
Paul Jolly: exactly yeah and
Cedric Charly: One
Paul Jolly: so
Cedric Charly: thing one
Paul Jolly: sorry
Cedric Charly: thing
Paul Jolly: please
Cedric Charly: before I hand it off. Um the the reason I originally know about this and I remember is reading the SR the Google SRE book and they talked about like in one example they brought up the concept of a configuration package for like some example they are bringing and at that time and this was like years ago I was like oh that's actually kind of obvious you know um so myself if you have any remembrance of that like I'm pretty sure it's pretty standard at Google to have configuration packages built by Blaze as part of like and that's kind of like version separately
Marcel van Lohuizen: Yeah.
Cedric Charly: to
Marcel van Lohuizen: Yeah.

00:48:06

Cedric Charly: the application versions. Yeah.
Marcel van Lohuizen: There there's multiple ways to do this. Uh you can also have a more dynamic sort of uh updating of configuration. So there's multiple configuration delivery systems and we've been thinking about some of those at least. So maybe maybe it's good to to take this uh offline indeed and have a more in-depth discussion because there's
Cedric Charly: Yeah.
Marcel van Lohuizen: many ways
Cedric Charly: I
Marcel van Lohuizen: and
Cedric Charly: will say there's a gap right now in our current open source tooling. I don't think there's an equivalent. So it could be something
Paul Jolly: Yeah.
Marcel van Lohuizen: I'm
Paul Jolly: Yeah,
Marcel van Lohuizen: still
Paul Jolly: and I think just to just to to briefly follow up on that point in terms of the gap in the open source tooling piece, just coming back to a point I made earlier on as to some of these things we will solve at a language level, some
Cedric Charly: Mhm.
Paul Jolly: at a tooling level, some
Cedric Charly: Right.

00:48:48

Paul Jolly: potentially actually require us to have some sort of, you know, actually running service or whatever somewhere that facilitates these things because actually it's not just a oneshot thing. it is a this thing is a longunning process in some way shape or form. So I think it's it's just trying to work out what the what the building blocks are in each of these places. Sort of then sort of saying okay yeah this kind of fits for all these different use cases that we've got in mind. Make sure are we solving all the problems that people have here particularly those human coordination problems which are just very real in this space. Um and uh and also things like security sensitivities etc. Right. There's quite a lot that needs to be fitted into this, but um uh Joshua Joshua Gilman pulled together a really good example that sort of started this motiv kickstarted the deployment patterns session back in February. So I look forward to actually re-engaging and uh on that side of things and hopefully sharing more of this stuff to encourage others to do the same as well and really generate a whole lot of ideas here.

00:49:52

Paul Jolly: Sorry, Steve.
Steven Harris: Just want to say mentioned some prior art the flux project recently as probably three and a half years ago started using OCI artifacts to do what I it's kind of like gitless githops. So instead of polling reading git repositories to find the kubernetes manifests that it needs to apply in the cluster uh they allowed you to publish OCI artifacts and stuff and so it just it's very much overlapping with what I think Cedric was talking about earlier with publishing configuration packages in that case you know if you consider the manifests are nothing but configuration um I was using that it was working Q was earlier in the process. So I used Q to generate the YAML and JSON that went into those OCI artifacts and was kind of done uh exporting and validating by the time it got published. But just then turns into a an artifact management problem like how do you keep too many of them from piling up in the registry things like that.
Cedric Charly: Yeah,
Steven Harris: So,
Paul Jolly: done.
Cedric Charly: I I've heard of GitOps before.

00:51:04

Cedric Charly: I think I didn't know the specifics on how they generate that into an artifact and then you know manage all the versions and map them to application versions if you need to switch it dynamically. I think that's where the complexity
Steven Harris: wow.
Cedric Charly: is.
Paul Jolly: Awesome, Steve. Thanks for the reminder on that. Yes, Stefan who wrote Flux also wrote Timone and Timone is um a friend of Q and Q is a friend of Timone.

Yeah, I I think there's in specific with specific patterns or specific um modes or systems that exist already. I absolutely think we can um learn from those and indeed you can as you do do as you suggest Steve and actually use Q to drive or and or sort of act as glue between parts of that system. That's that's absolutely clear. I think the key thing um uh yeah Roger just hinted at one thing there and the other thing that I was going to say is one of the key things to my mind is the from a Q perspective the this whole concept of having um configurations or values that are incomplete that are later supplemented by values in a very principled way um and also this concept of allowing teams to coordinate at a language level that's something that fundamentally is something that Q supports and again via especially the concept of defaults but also incomplete values as well.

00:52:24

Paul Jolly: So it's just making sure that we when we're thinking about things from a Q perspective, especially the language level, but also the primitives on top, that actually we're learning from things like Flux and other systems in terms of what works well, but also what doesn't work quite so well, for example, where you haven't got good opportunities for coordination or conflict resolution between teams. And so it's making sure we've got those examples either specifically from things like flux or just sort of slightly more abstractly we we need to coordinate between these three teams here on getting this configuration here. This team has priority etc. We're happy with specific examples as well as abstract this is how we think we want to do it type things and then trying to map that to that space. So yeah I definitely think we can learn from flux but I don't mean to sort of shorten the list to just flux. we can learn from any system. So again, if people have good experience of other systems, absolutely please include that in those experience reports as well.

00:53:29

Paul Jolly: Okay, great. In which case Oh. Oh, please Steve again. Yeah.
Steven Harris: Sorry, got in
Paul Jolly: No,
Steven Harris: a little
Paul Jolly: don't

Steven Harris: late.
Paul Jolly: us.
Steven Harris: You may be able to tell from that massive thread that I was driving in Slack a couple weeks ago. Um, I have renewed interest in in kind of air gapped use of modules. So, you could also say that it's kind of like the vendoring problem, but um, the Q tool is very clever right now. It's kind of late on demand. It will fetch modules and cache them. And it's it's really nice that it does that. doesn't force you to think ahead of time about how to use modules or steps to start consuming them. But my particular interest and maybe on the call might all be interested is in Basil and Basil's model kind of splits things in time so that um by the time the Q tool runs to use modules I need to think about it as kind of being air dapped um in the sense that I'd prefer that it not rely on network at that point to be able to find all the module related files in a particular place.

00:54:38

Steven Harris: So, I almost have all that I need, but you know, we've got a little bit of a tooling gap that we have some open issues addressing. There's the the idea, I think it's Q mod download is one of the proposed uh sub plans. Yeah. Um, so I just just want to say I'm I'm still interested. I don't know if you're looking for external help on that or if anyone else has any other um use scenarios that they want to discuss, but that's my motivation to do that.
Paul Jolly: That's kind Roger. I
Marcel van Lohuizen: on
Paul Jolly: don't know whether
Marcel van Lohuizen: up.
Paul Jolly: you want to say anything in particular there. We also have Q mod vendor as well. Um
Roger Peppé: both
Paul Jolly: there's
Roger Peppé: of those
Paul Jolly: something
Roger Peppé: are
Paul Jolly: about space.
Roger Peppé: on on the road map. Um there's a a bit I mean I suppose the main the main um the main thing is just finalizing what we want the design to look like in terms of what what that those command line because a basic version is really easy but but do

we want it to actually look like that?

00:55:33

Roger Peppé: if you want to commit to that uh you know basic design. Yeah.
Paul Jolly: Steve, I think it would be good on either QOD download or especially QOD vendor if you've got any specific requirements from your side um just chuck them in there because I think
Steven Harris: Okay.
Paul Jolly: honestly it it's um an open source project that is based on top of Q ex also wants Q mod vendor as well um and
Steven Harris: You know,
Paul Jolly: um their use case might be slightly different from what you need for basil as it were. So please get the basil requirements in there noting our exchange on the embed side of things where from a sim link perspective that you're coming at it from a a specific set of constraints for one of a better phrase on the basil side of things. So please do actually um uh chuck that in there. The
Steven Harris: We'll
Paul Jolly: the whole thing again just to briefly mention it about about the air gap the airgapped requirement more generally again this was like a key consideration in using the OCI model.

00:56:37

Paul Jolly: So whether for basil or a truly airgapped say for example like sensitive for whatever reason military whatever you might consider it scenario or just I've unplugged my computer from the network scenario is to be able to do this airgapped approach. Um, and so the fact that we have OCI artifacts is a key part of that from just being able to mirror OCI artifacts between different servers which support that air gapness or as you said literally just using local caches whether it be a visible vendor or a hidden I've downloaded everything to the cache. Um, again so that that's kind of where I'm trying to nudge you. If you want to see the the the the vendor then it's QOD vendor. If you want to sort of just pull things out of the cache, it's going to be Q mod download and then they're subtly different. So if you can chime in on what you want and what you need, we've had if you like slightly more votes for Qod vendor on the basis that people actually

want to see see the vendor as it were in some way, shape or form. But you can absolutely achieve what you want already today by doing a bit of hackery um and do like a a poor man's Q mod download into an empty um Q cache deer. But at the same time is more kind of what do you need and make sure that as Raj said then we get the UI UX piece of that right from the command line. That would be great.
Steven Harris: Okay, I'll do that. it's
Paul Jolly: Which case we're a time. If thanks everybody for joining. As please do just ping us offline or just in slack or discord. If there's anything that we've missed and you'd like to see cover next time otherwise we'll post the recording and the slides and transcript as soon as they're done. Thank you, everybody. See you
Cedric Charly: Take
Paul Jolly: next time.
Steven Harris: It's much.

Transcription ended after 01:02:53