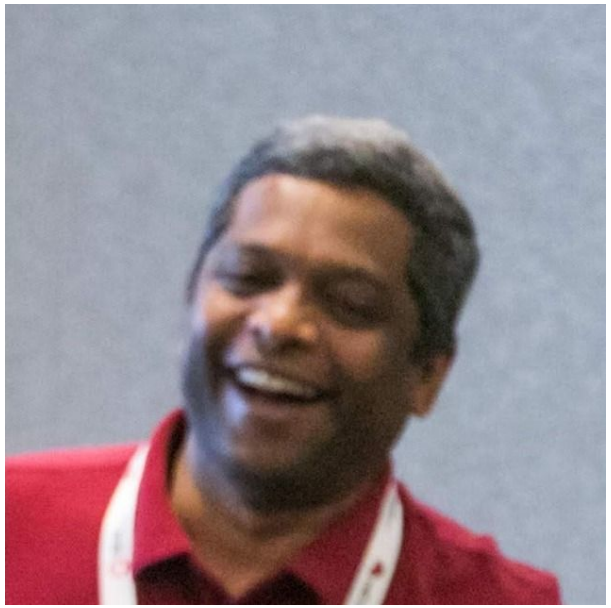
Kubernetes SIG Architecture Intro and Update

May 20, 2022

Who Are We?



Davanum Srinivas
VMware
@dims

John Belamaric
Google
@johnbelamaric

Goals of the Kubernetes project

- Portable
- General-purpose
- Meet users partway
- Flexible
- Extensible
- Automatable
- Advance the state of the art

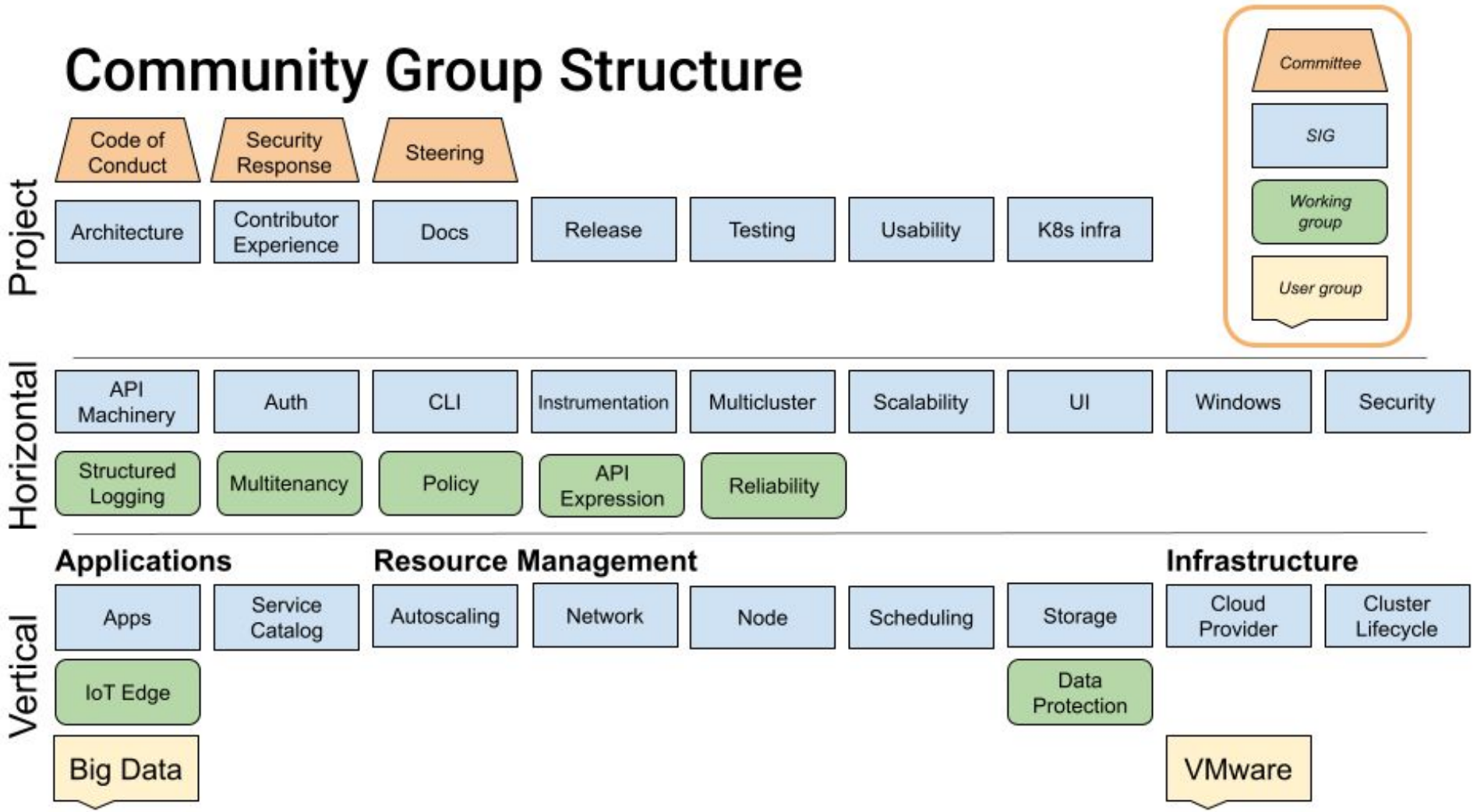
https://git.k8s.io/community/values.md

Kubernetes Community Values

- Distribution is better than centralization
- Community over product or company
- Automation over process
- Inclusive is better than exclusive
  - Your feedback is solicited
- Evolution is better than stagnation

https://git.k8s.io/community/values.md

Kubernetes Project Overview



SIG Architecture Scope

The Architecture SIG maintains and evolves the design principles of Kubernetes, and provides a consistent body of expertise necessary to ensure architectural consistency over time.

- *Conformance test definitions*
- *API conventions*
- *Architectural renderings*
- *Design principles*
- *Deprecation policy*
- *Production readiness criteria and reviews*
- *Kubernetes Enhancement Proposal (KEP) process*

Cross-cutting Processes

- Conformance test review and management
- API review process
  - go.k8s.io/api-review
- Design documentation management
  - git.k8s.io/enhancements/keps
- Deprecation policy management
  - k8s.io/docs/reference/using-api/deprecation-policy
  - k8s.io/docs/setup/release/version-skew-policy
- Production Readiness Reviews
  - git.k8s.io/community/sig-architecture/production-readiness.md
- Kubernetes Enhancement Proposal process

What other kinds of issues?

- Ambiguous behavioral questions
  - Inconsistencies in behavior across resources
- Unanswered questions
- Anything where TL/Chairs/Owners conflict
  - Not as escalations
  - Formulate general guidelines/principles
- Start a mailing list thread - come with KEPs and details!
  - git.k8s.io/community/sig-architecture#contact

Sub Projects

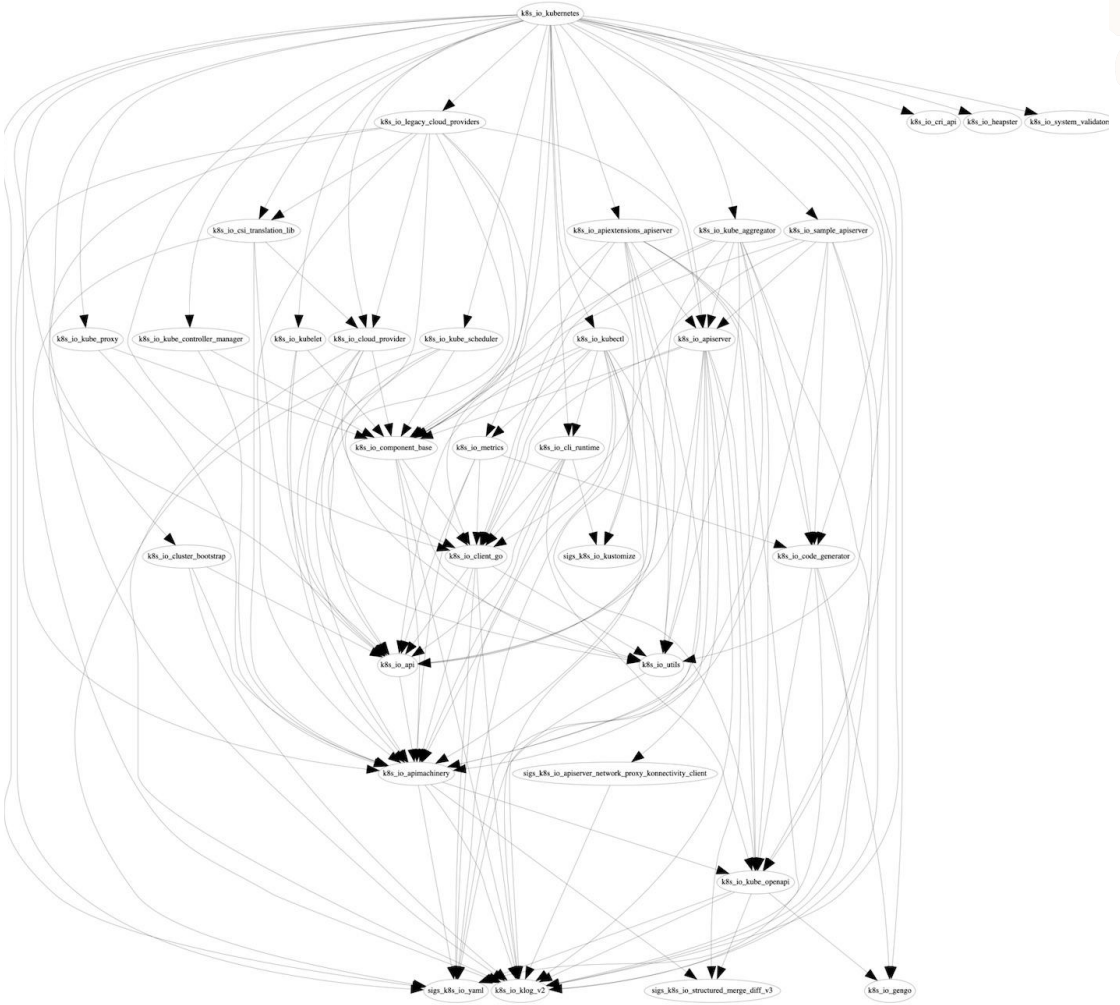
- Architecture and API
  - Document design principles
  - Document and evolve system architecture
  - Reviewing, Curating extension patterns
- Code Organization
  - Repository structure, branching, vendoring
- Enhancements
- Conformance Definition
  - Review, approve changes to conformance test suite
- Production Readiness Reviews

API Review

- Review process
  - go.k8s.io/api-review
- Project board
  - github.com/orgs/kubernetes/projects/13
- API Conventions, Guidelines
  - git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md
  - git.k8s.io/community/contributors/devel/sig-architecture/api_changes.md
  - Very relevant for in-tree API design / additions / changes
  - Some guidelines also apply to CRD development

Code Organization

- bit.ly/sig-architecture-code-org
- github.com/orgs/kubernetes/projects/27
- Dependency management
- Subrepo structure



Source: http://issue.k8s.io/76395

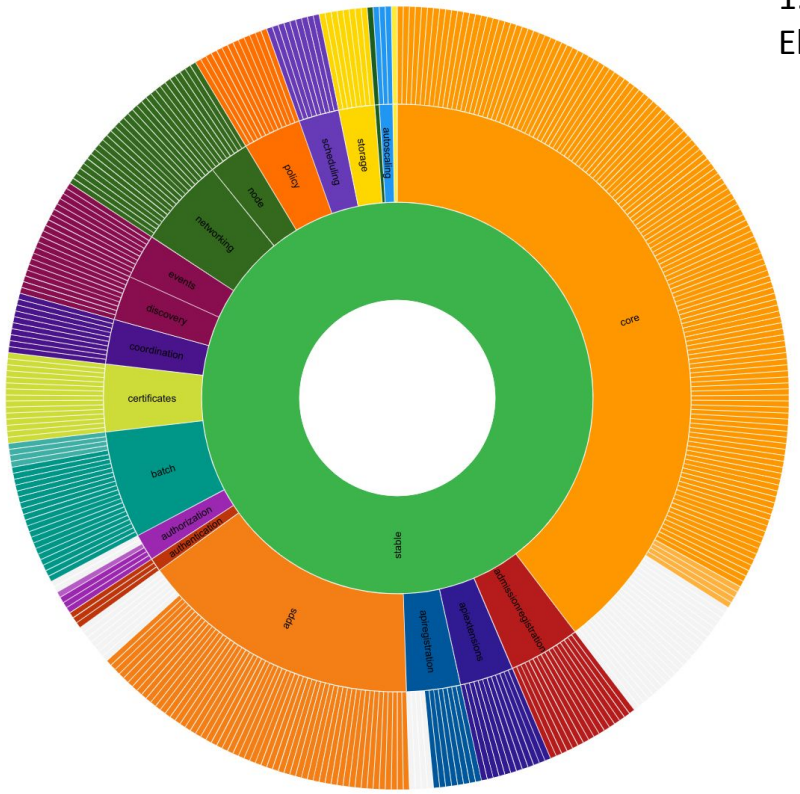
Enhancements

- Define and tweak process for KEPs
- Shepherd community members through KEP lifecycle
- Automate steps when possible
- Work with SIG-Release team on release boundaries
- Make it easier to find information and keep things up-to-date

Conformance Test & Promotion

- Ensuring consistent support and behavior across distributions
  - bit.ly/sig-architecture-conformance
  - github.com/orgs/kubernetes/projects/9
  - git.k8s.io/community/contributors/devel/sig-architecture/conformance-tests.md

- Visualizing current coverage
  - apisnoop.cncf.io
  - Filter by stable/beta/alpha status
  - Filter by API group
  - Filter by test



1.24 Coverage of Conformance Eligible Endpoints

406 total endpoints
91.13% tested (370 endpoints)
88.92% conformance tested (361 endpoints)

Production Readiness Reviews

KubeCon CloudNativeCon Europe 2022

- bit.ly/sig-architecture-prod-readiness
- Asking the question "how will people run this in production?"
- Feedback loop from cluster operators, features that went well / didn't go well
- Developing questions/processes to improve production readiness
- Examples: monitoring, admin documentation, rollout, scale, security

Cluster Operators

Please fill out our survey today!

https://forms.gle/mabABXgcYdM79dHw5

Extensions - Design

- Avoid bottlenecks
- Increase supported use cases / capability
- … without increasing complexity
- Permission/blessing not needed
- Rapid iteration
- Distribute responsibility
- Capture best practices / consistency
- Guard rails around integrations

Extension - Examples

- API aggregation -- multiple api servers
- ThirdPartyResource/CRD/Storage -- key/value CRUD + metadata
- Admission-control extension web hooks -- pluggable policies
- Scheduler Extensions
- Built-in add-on manager -- reconcile configuration & self-hosted components
- Component registration
- Kubectl extension
- Component configuration (via ConfigMap)
- Container runtime (CRI), network plugin (CNI), volume plugin (CSI)
- External cloud provider
- External secret management (KMS)
- External controller pattern
- Extension schema validation and discovery

Where are we going?

- Focus on extensions
  - Let the ecosystem grow and distill the best patterns
  - Fewer changes to core
  - Prove out new ideas as CRD APIs
- Build out conformance
- Cleanup + Reliability (.0 releases, production readiness)
  - features *must* go GA effort
  - Ensuring follow through
- Organization Scaling
- KEPs KEPs KEPs

How you can participate?

- Attend the main and subproject meetings
- Follow along on project boards, mailing lists, and slack:
  git.k8s.io/community/sig-architecture#contact
- Find something of interest you can help with
- Speak up - offer your thoughts and ideas,
  ask questions for background/history, etc.
- Help with issue triage, PR reviews, docs

THANK YOU

Q&A