Proposal: Istio Foundational Mode

Over its almost 6 year history, Istio has amassed quite a wide set of features and configuration toggles for customizing the behavior of a mesh. However, many of these user touchpoints have varying feature/supportability statuses, and users who only want to consume production-ready features must survey all desired features *and how they're represented* within the control plane (i.e. CRD vs. meshConfig vs. env var) as well as develop bespoke enforcement mechanisms (e.g. gatekeeper, custom validating webhook etc). As such, it is desirable to create a "foundational mode" for Istio so that users can have confidence that only production ready code paths will be executed. Furthermore, this new operational profile will provide Istio maintainers with a way to be opinionated about what features users should be consuming.

Goals

- Must be configurable via Helm chart
- Must reject/warn on configuration that violates "foundational supportability level" at runtime
- Document surface area for features
- Improve feature status hygiene and maintenance burden
    - foundational mode means we need to get better at promoting features to beta when they reach that level of maturity
    - Should be reviewed each release

Prior Art

Kubernetes Feature Lifecycle + Gates

Kubernetes features have a defined lifecycle that they iterate through. All alpha features are disabled by default, and most Kubernetes providers do not allow a mix between alpha and non-alpha features within the same cluster. Kubernetes also has many feature gates users can turn on/off to enable specific features in a given cluster. Note that there are some proposed changes to this process from Tim Hockin: Thinking about alpha/beta/GA in k8s (public).

Principles for "foundational mode" inclusion

The following principles are meant to serve as guidelines for whether or not a particular feature should be included in "foundational mode". It is strongly encouraged that these principles remain

immutable once agreed upon by the TOC; however, the way in which these principles are implemented may change with TOC approval.

1. Istio “foundational mode” prioritizes “safety of the user” above all else
    a. The definition and practical benchmarks for “foundationalty” are malleable, but generally they should include stability, performance, test coverage, and developer experience.
2. Only features beta and above should be included in foundational mode with exceptions requiring TOC approval
    a. Any exception must have an Istio community member as an owner, be sponsored by a WG lead, and have a clearly documented plan for reaching beta including an ETA. If that ETA is not met, the owner and WG lead must submit a revised plan and ETA.
    b. Features that are exceptions can be removed from foundational mode with a TOC vote
        i. In these scenarios, the TOC is encouraged to consider other factors such as adoption of that feature and overall project health
3. There should ideally only be one way to accomplish a task in “foundational mode” and that way must be at least beta.
    a. If at any time there exist two “features” in foundational mode that do the same thing, the TOC should remove one of them. In these scenarios, the most future-forward feature should be kept
4. Any removal of a feature from foundational mode requires advance notice of 4 releases
    a. Note: foundational-mode removals should not be marketed as “deprecations”. The TOC is not taking a stance saying that the feature should not be used but rather that it no longer meets the criteria of being “foundational”
5. Any beta feature in foundational mode must have an ETA for reaching stable that is reviewed by TOC

Intended Audience for “foundational mode”

- First time users who need a better “batteries included” experience that leads them towards best practices
- Platform engineers/SREs who are building tools on top of Istio that need relative stability and future-proofing when it comes to the de-facto APIs and installation methods (e.g. IstioOperator and Gateway API)

Ambient vs. foundational Mode

Ambient mesh and foundational mode are both initiatives within the Istio project to promote long-term stability and production-readiness. Ideally, the two initiatives will GA ~the same time

with Ambient being a part of foundational mode along with sidecars as deployment topology options.

Action Plan

1. Ensure feature list is complete
2. Create a doc correlating that feature list with the relevant configuration options and where they live (MeshConfig? CRD? Env var? Binary flag?)
    a. We should also clarify if there are inherent support levels for different enablement mechanisms (e.g. env vars are inherently experimental)
    b. Also, identify features that need promotion to a different mechanism (e.g. move a setting from an env var to a meshconfig setting or API)
    c. Will be tackled via https://github.com/istio/enhancements/issues/146
    d.
3. Create a design doc for the runtime enforcement mechanism in light of the above enablement mechanisms
4. Create a doc/parent issue for the various installation configuration changes necessary for foundational mode
    a. This is where we'll decide on separate helm chart or not
5. Implement
    a. Install mode without any untested options in the helm chart
    b. CRD subset (no EnvoyFilter, etc)
    c. Istiod validations for unsupported features
    d. Warnings/metrics for existing features in cluster

Proposed Policy Changes

1. All env vars are inherently experimental (and should be documented as such)
    a. If env vars have been defaulted to true for 4 consecutive releases (more? less?), that env var should be removed
2. Something about meshconfig
    a. It's not well documented, but meshconfig actually has versioning in istio/api. There is only a single version (v1alpha1); is there a future where these move to beta?

Execution options

Foundational Istio is a major change and likely will take a few releases to complete. It is highly desirable to provide incremental progress and gather early feedback. While "foundational Istio mode" will focus on beta and stable features - the "foundational Istio" should be considered first

as an experiment, alpha - and be ready to make changes based on feedback before moving to beta.

The ‘action plan’ is the long-term requirement, but it can be done iteratively.

A first step could be a trimmed down helm chart that doesn’t allow overrides and only includes the options that are part of the CI/CD test suite. Istiod would also have a ‘foundational mode’ option that will disable reading the alpha CRDs and restrict what changes in MeshConfig are loaded, as well as annotations not on the ‘beta’ list.

Defining an install ‘profile’ for istioctl is not ideal - it is desirable to have a single install mode for ‘foundational’, and Helm has better integrations with Terraform and adoption in many tools. We can avoid divergence from the main Istio chart by making a copy of the unmodified files and just changing the values.yaml - but the injection templates are full of special cases based on annotations/pods with minimal testing.

Timeline

Because the 1.18 feature freeze is on 4/11, we’re targeting the 1.19 release to debut Istio foundational mode.

Intentional Drift

Proposed API surface

Note: this new mode is itself alpha as we refine the criteria and features within it.

Traffic Management

- Gateway API: Ingress, Egress, Mesh for all protocols
- Protocols: HTTP1.1/HTTP2/gRPC/TCP, Websockets, HBONE
- Resilience features: timeouts, retries, connection pools, outlier detection
- TLS termination and SNI Support in Gateways
- SNI (multiple certs) at ingress
- Locality load balancing
- Sidecar API
- …

Security

- PeerAuthentication
    - STRICT by default
- AuthorizationPolicy
- Pluggable CA/Cert
- Ingress Gateway cert support
- Auto-mTLS (via HBONE)
- …

Observability

- Prometheus Integration
- Distributed tracing (sinks tbd)
- Trace sampling
- …

Other

- WorkloadEntry/WorkloadGroup
- ServiceEntry/Egress
- Basic Helm install options
    - Repo and tag
    - Common helm install overrides (CPU, platform-specific, etc)
- Istio CNI plugin as default - and initially as the only option (sidecar with elevated permission is not the foundational option - we may add it later but best to pick the best option when multiple choices exist)
    - Note: excludes users running with non-chainable CNI
- Multicluster mesh
- …

Notable Omissions

- Subsets
- Envoy filters
- VirtualService
- Label based routing
- MongoDB protocol
- ALPN-based auto mTLS
- IstioOperator (why is this beta on the features page??)
- MCS (for now)