Enterprise Standards

February 2022

Gary Valley
Chief Software Architect (SAIC)

Defining Enterprise Standards

Policy: A statement (typically from leadership) defining an intent or organizational purpose.

- Example: The legal team will review all contracts prior to signature
- Mandatory for the intended audience

Standard: A specification of uniform technologies or configurations to solve particular problems or achieve specific efficiencies

- Example: Ground-based software systems use TCP/IP for information exchange
- Voluntary (unless explicitly mandated through policy)
  - Allows for flexibility
  - Creates cost for non-participation
  - Network effects – there's little use in creating technology if it doesn't interoperate with other technology
- The Department of Defense (DoD) has a propensity to mandate standards (via senior leader memoranda)

Guideline or Best Practice: Advice or support to set example or expected behaviors in relation to policy

- Example: Try to understand the mission context prior to developing systems
- Always Voluntary

Today's discussion focuses on Software APIs

Forming Standards – Domains

Battlespace Awareness

- Space Domain Awareness
- Weather
- Intelligence Data
- Indications and Warnings
- Sensing

Planning

- Orders Management
- Course of Action (COA) generation and optimization
- Modeling and Simulation

- Subject Matter Experts (SMEs) for different components of the mission (i.e. “domains”) are different
- Data formats and software APIs across domains likely change at variable rates
- Allowing leadership to choose the priorities for advancement of any phase while maintaining interoperability is critical to mission success
- Capability development (i.e. advancement/innovation) seems at odds with interoperability (i.e. communication across domains)

Maintaining Interoperability **AND** Rapid Capability Delivery

- Numerous inter-domain interfaces could potentially change if we allow for continuous capability development within domains.

  Services implement versioning and deprecation patterns

  - Versioning
    - Ensures configuration management teams are delivered tested and labeled software services.
    - Improves interoperability by ensuring that tested configurations remain stable for a multi-component software version configuration
    - Allows development teams to continuously deliver capability, exposing “upcoming” technology to other software teams
  - Deprecation
    - Allows for older versions of software to be removed from the maintenance burden
    - Improves security by reducing the number of exposed interfaces
- Standards are derived from the publication of these versioned, deprecated, tested APIs; produced from the specified domain
  - But where are these standards published?

Publishing Standards – Service Registry

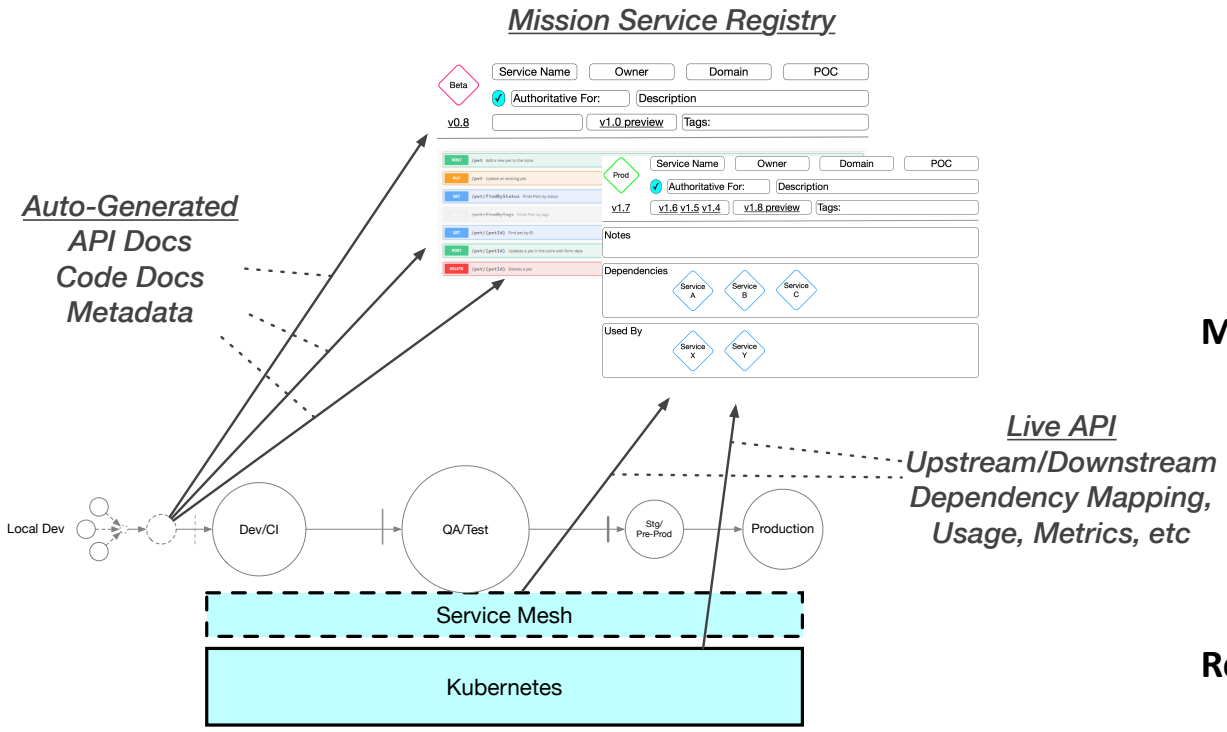

Diagram courtesy of Ben Adams (SAIC)

Service registry provides a unified UX/UI for developers to browse and search for applications or services

- Hosting the active versions of software with associated documentation and support metadata (e.g. Author, release date, etc.)
- Provides the ability to solicit usage and manage delivery
- Share future capabilities or features

Monitoring of service health and availability

- Reporting new or deprecated services
- Metrics on existing services (i.e. requests per minute, uptime, etc.)
- Dependency tracking – determining critical services

Reduces the potential for duplicative development efforts

Upfront Access to documentation

Text courtesy of Jorge Rodriguez (MITRE) and Steve Baek (MITRE)

Services must publish interface documentation, version, and testing information to a mission service registry

Publishing Standards – Service Registry

Mission Service Registry

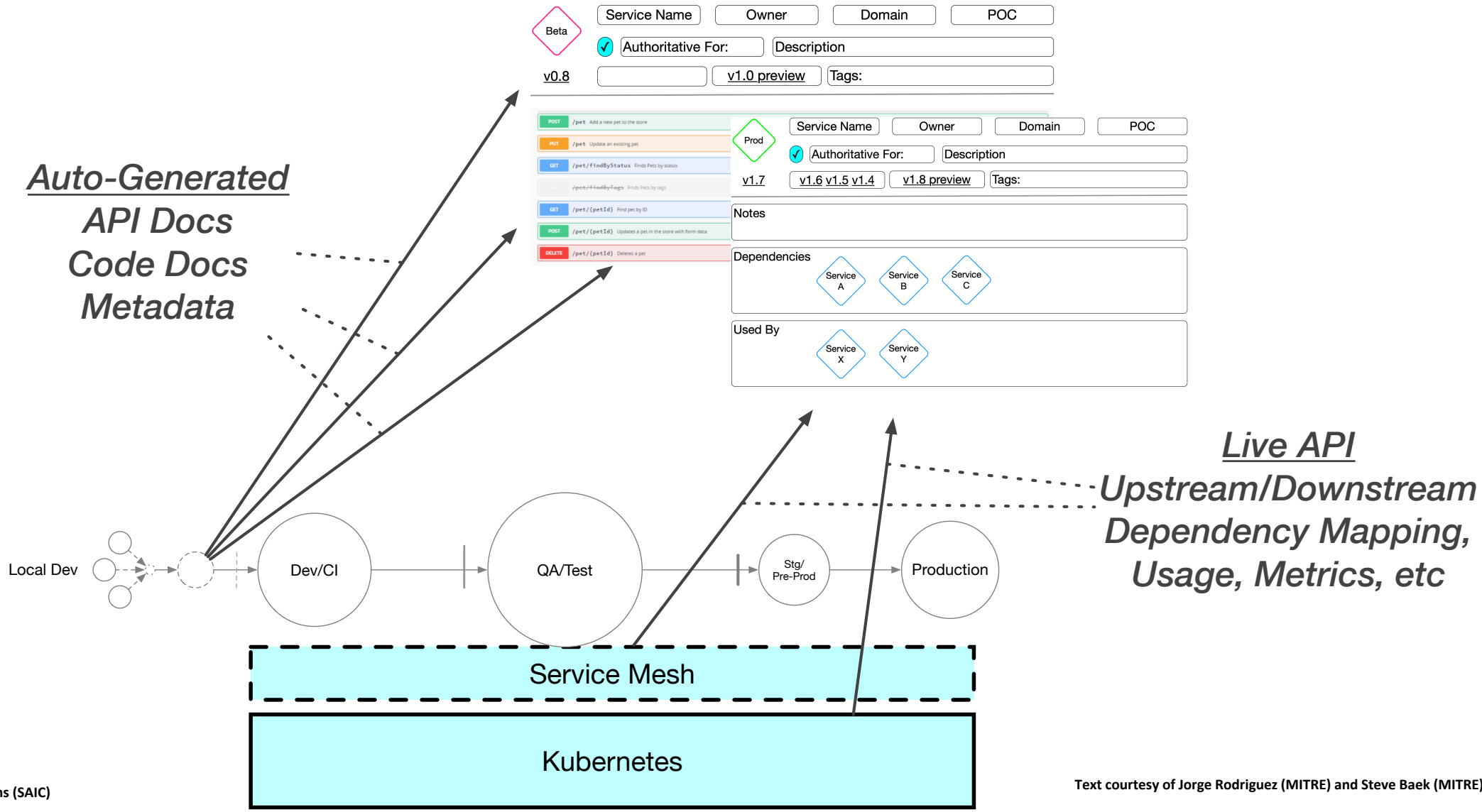

Diagram courtesy of Ben Adams (SAIC)

Text courtesy of Jorge Rodriguez (MITRE) and Steve Baek (MITRE)

Transitioning to Domain Driven Standards

The Space Enterprise must define the pertinent mission domains

- What are the domains critical to the success of the space mission set?

Organize the space enterprise around those domains

- Ensure that subject matter experts work within their domain of expertise. Drive the knowledge of software engineers for that domain to develop cross-communication between SMEs and software engineers.

Create best practices to ensure versioning and deprecation patterns and publication to mission service registries

- Drive for self-service: Developers external to a domain should be able to reliably discover and consume interfaces in other domains

Automate the production of documentation

- Automated injection of dependencies, documentation, testing results, etc. lowers the burden on developers
- Reinforces best practices

Contact Information

Thank you!

Chief Software Architect (SAIC)
Gary Valley
gary.valley@saic.com