Get Ready for a New Kubernetes CD Pipeline

What you will be building to support CD in a microservices architecture

Steve Taylor, CTO DeployHub

DEVOPS WORLD by CloudBees

Takeaways

Re-imagining CI – Builds go away for the most part. Linking is done at runtime.

DevOps Scaling – We are transitioning from managing a single application to hundreds of independently deployable services and components. Scaling will be a challenge.

Mono Repo Vs. Poly Repo - For most organizations, microservices will have their own repository and their own CD Workflow.

New Stuff – Domain Driven Design, Service Mesh and container versioning will mature to be a part of the CD process.

DEVOPS WORLD by CloudBees

Meet Steve Taylor

Passionate About Putting Things Together – From Software and Beyond

- CTO and Co-Founder - DeployHub, Inc.
- CTO and Co-Found - OpenMake Software
- 20+ DevOps Experience
- Volunteer Fire Chief

DeployHub®

Continuous Delivery Defined

Core to the DevOps Movement

"CD is a software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time. The rise of microservices, cloud native architectures has caused a corollary rise in continuous delivery practices. This is related to CI/CD that includes Continuous Integration (CI) -- the practice of merging all developer working copies to a shared mainline several times a day."

CD.Foundation

In the Beginning...

The CI Build – Let’s review what CI is for.

The All-Important Binary Object

The CI Step:

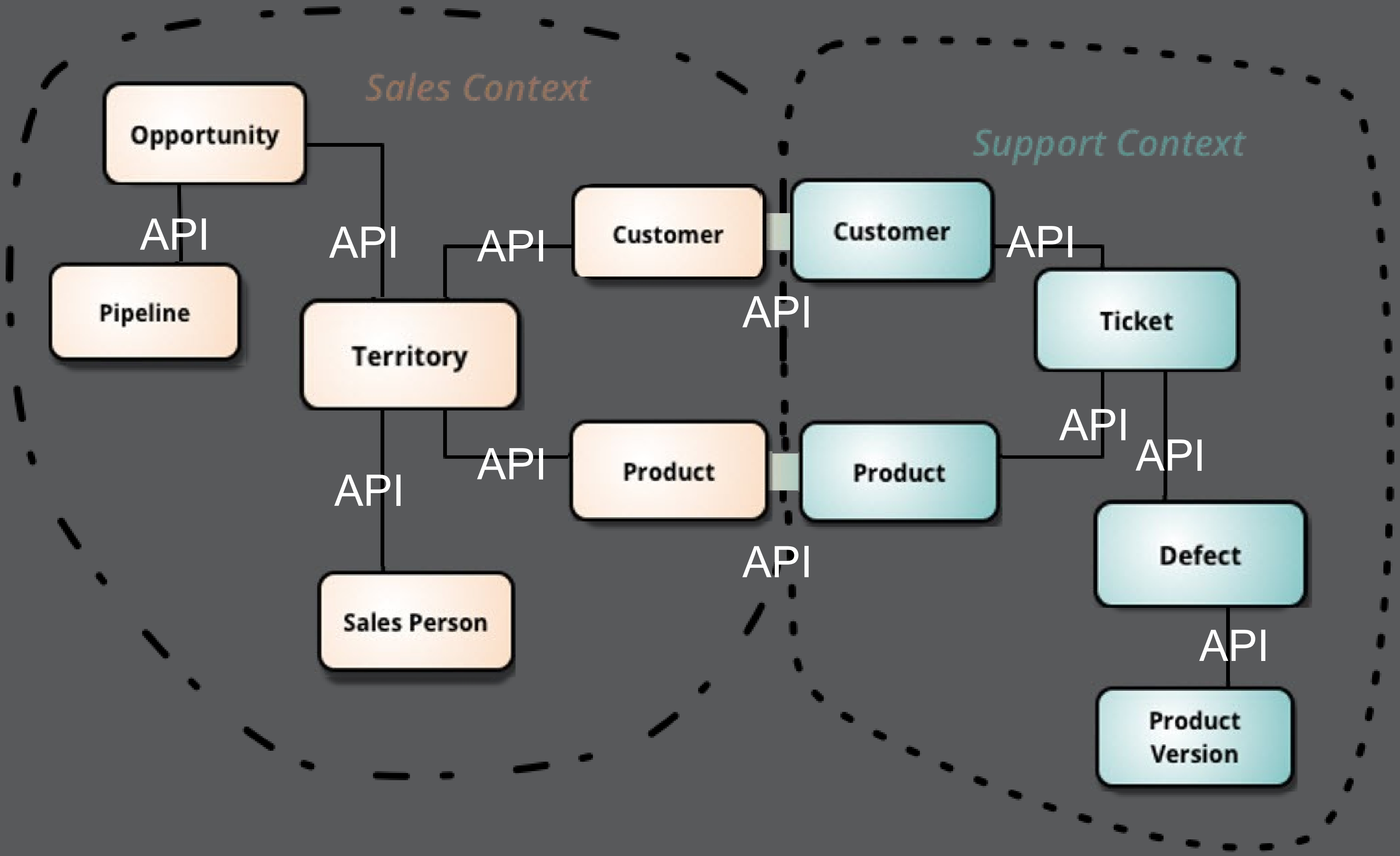
- Merge/Pull Code
- Compile/link the Application binaries
- Generate BOM
- Track Differences
- Execute Deploy Script

The End of the Build

Version Control – Less branching and merging.

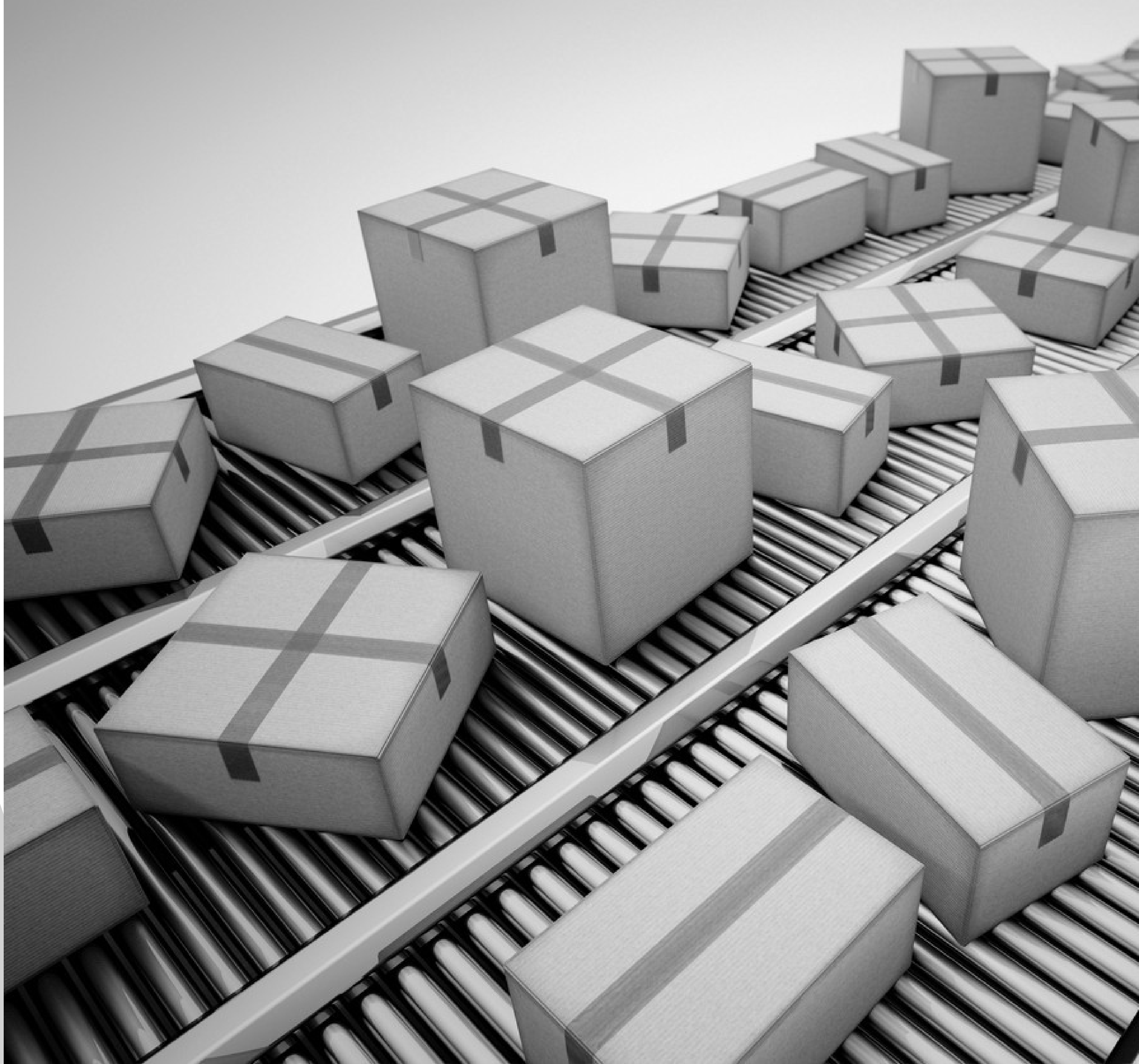
Builds create and register a docker image.

Small compiles with little to no linking – if at all. Think loosely coupled.


Sales Context
Support Context
Opportunity
API
Pipeline
API
Territory
API
Customer
API
Customer
API
Ticket
API
Product
API
Product
API
API
Sales Person
API
Defect
API
Product Version

DevOps at Scale

“Independently Deployed” is code for “fast and frequent.”

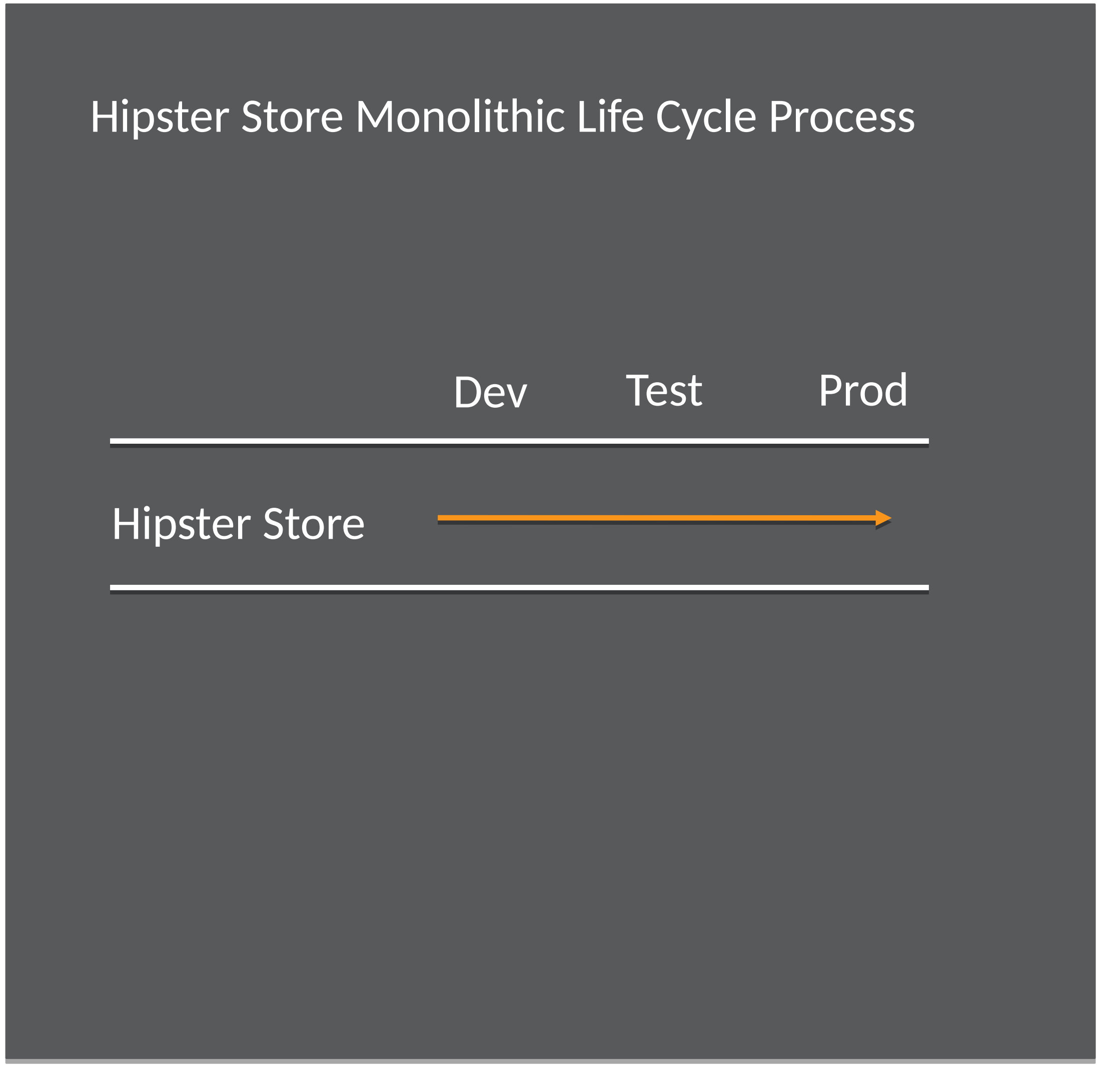


Monolithic Lifecycle

Static Application once created, never re-built.

One workflow.

Planned daily, weekly, monthly deployments.

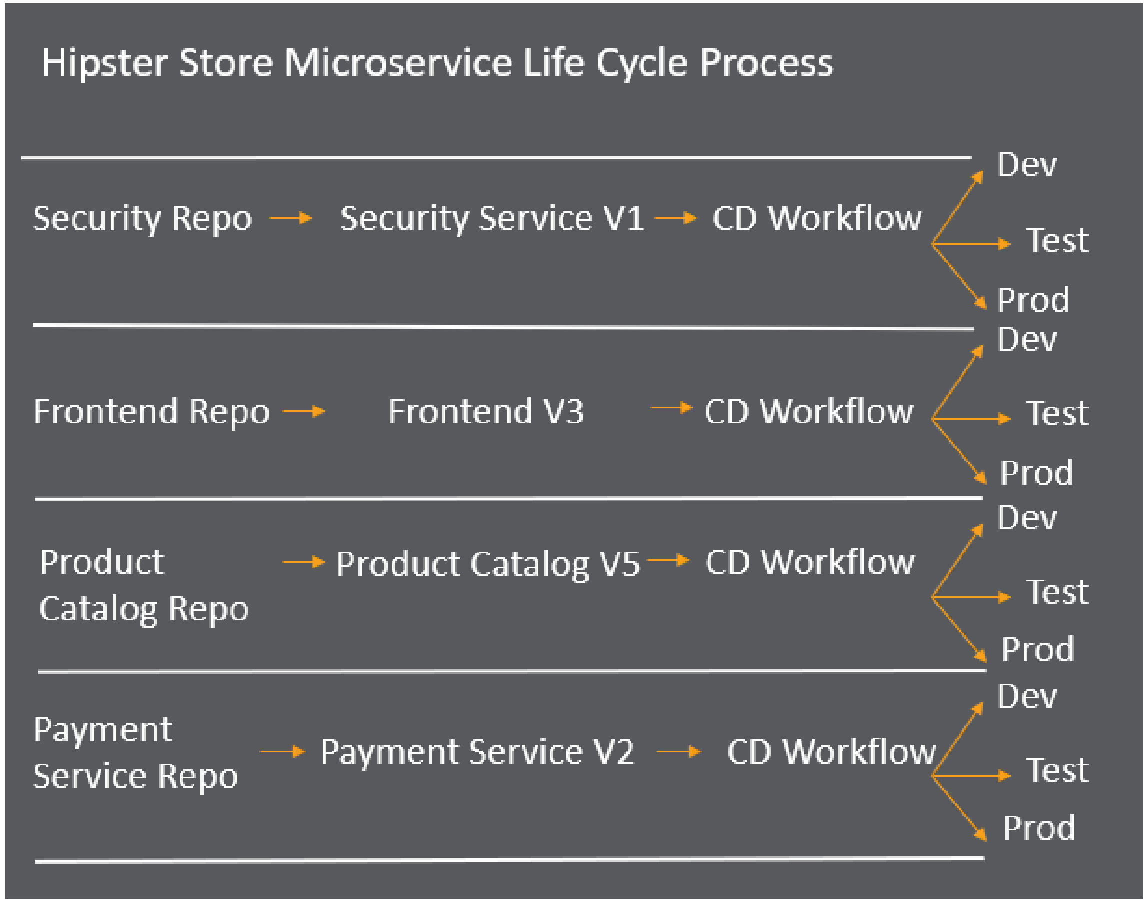


Microservice Lifecycle

Dynamic independently deployed services.

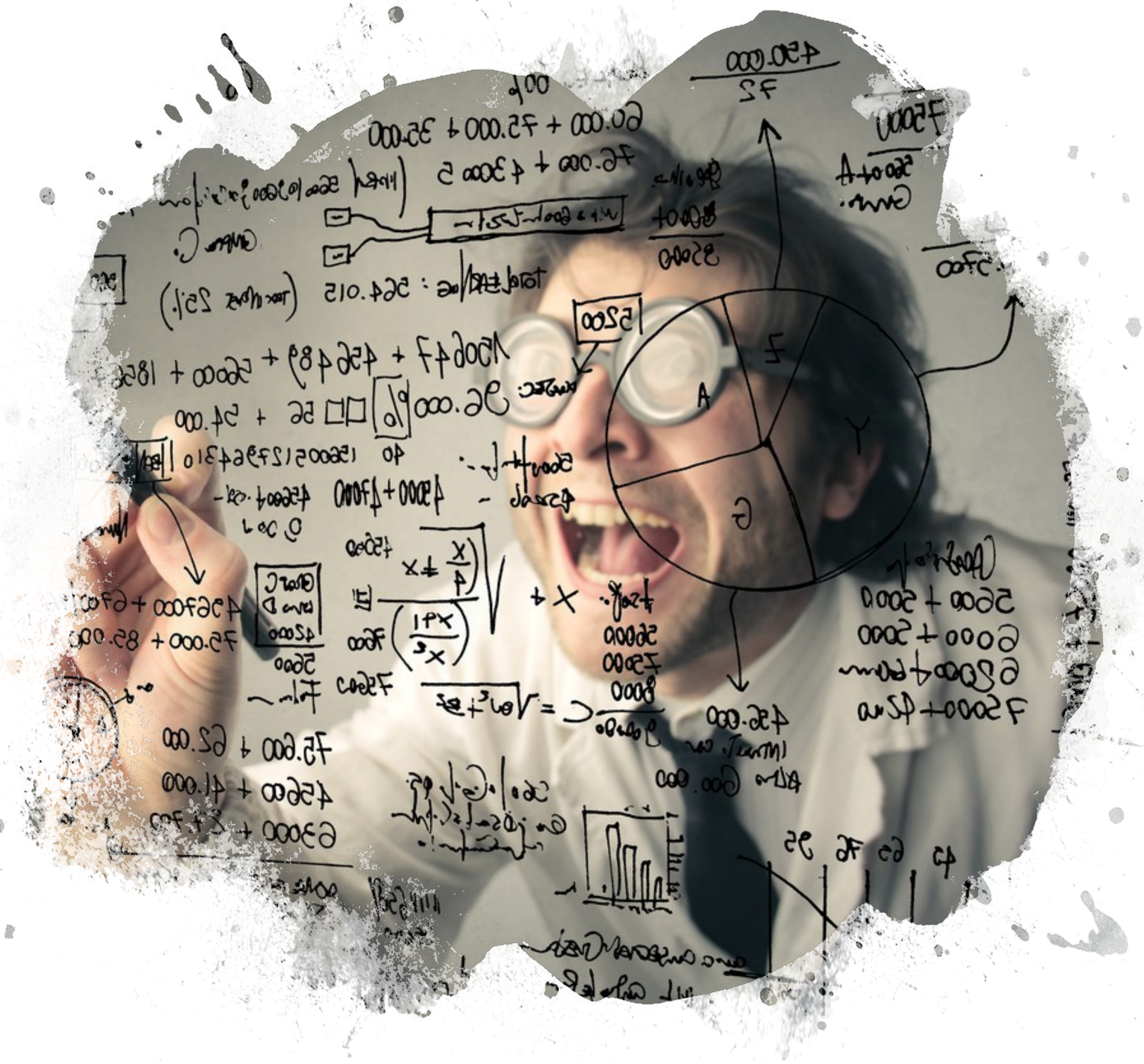
Lots of workflows.

Deployments all day long.



The Decline of Scripts

Separate the Data from Definition to support each state in the lifecycle.



Scaling CD

- Templates, Events and Custom Resource Definition

The Repo Balance

Mono Vs. Poly

Mono Vs. Poly

- ✓ Mono Repositories minimize then number of CD Workflows.
- ✓ Security is contained to a single repository.
- ✓ No Coordination across repositories.
- ✓ All changes follow the same path.

- ✓ Poly Repositories supports independent CD workflows and releases.
- ✓ Finer grained security on subset of repositories.
- ✓ Minimize branching and merging.
- ✓ Different Workflow Paths for Service.

New Stuff

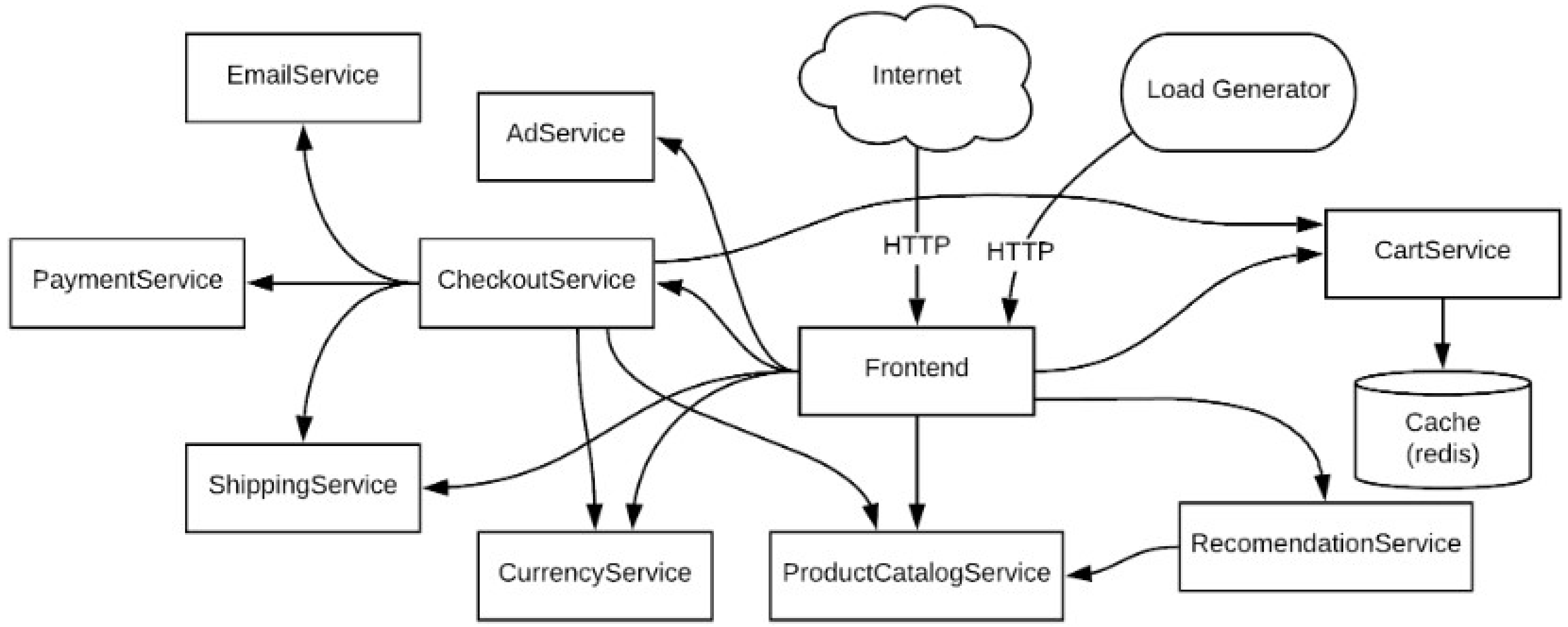
Domains

Service Mesh

Container Versioning

Domains

Domain Driven Design

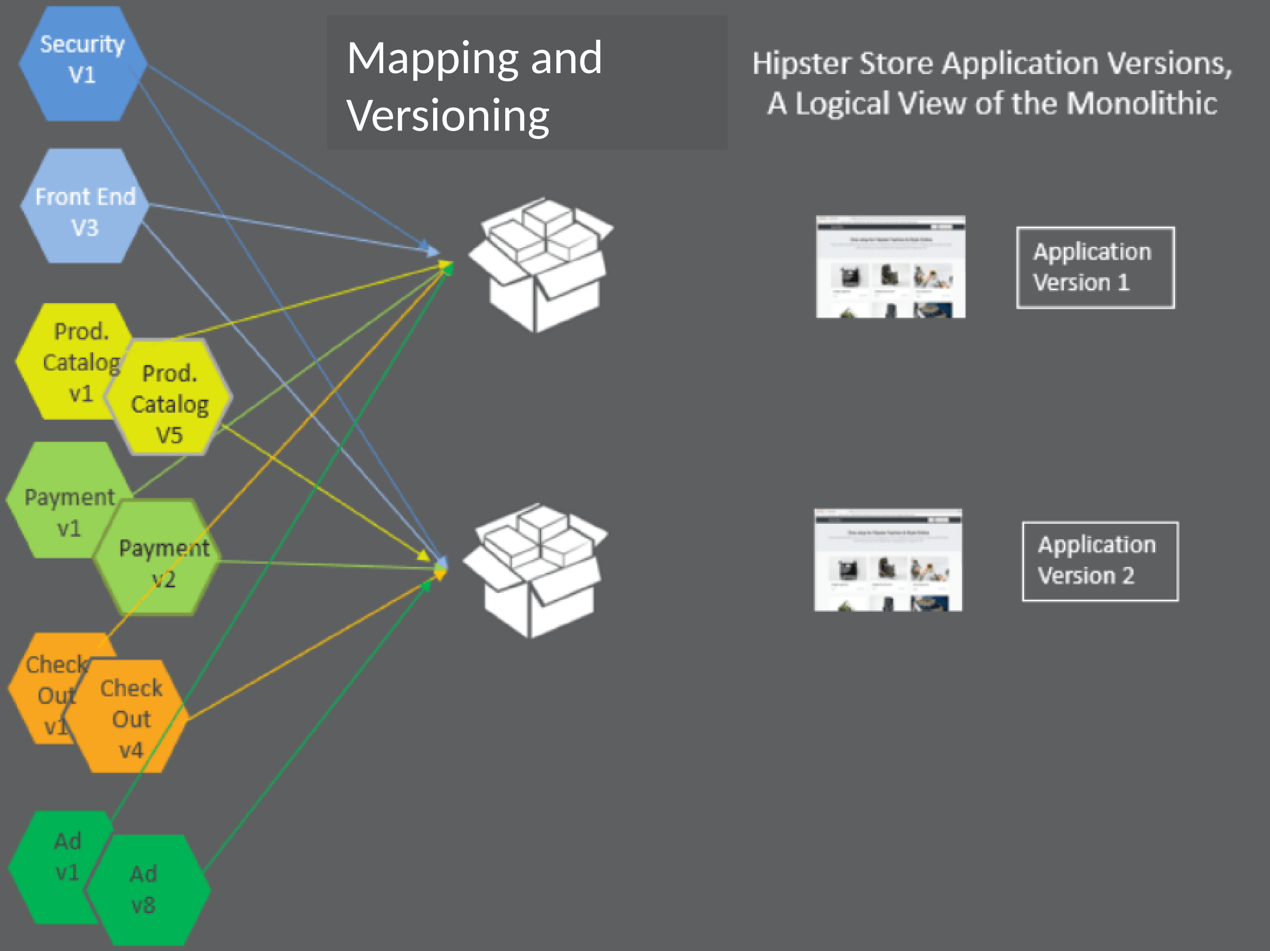


Container Versioning and Service Mesh

Service Catalog

Mapping and Versioning

Hipster Store Application Versions,
A Logical View of the Monolithic

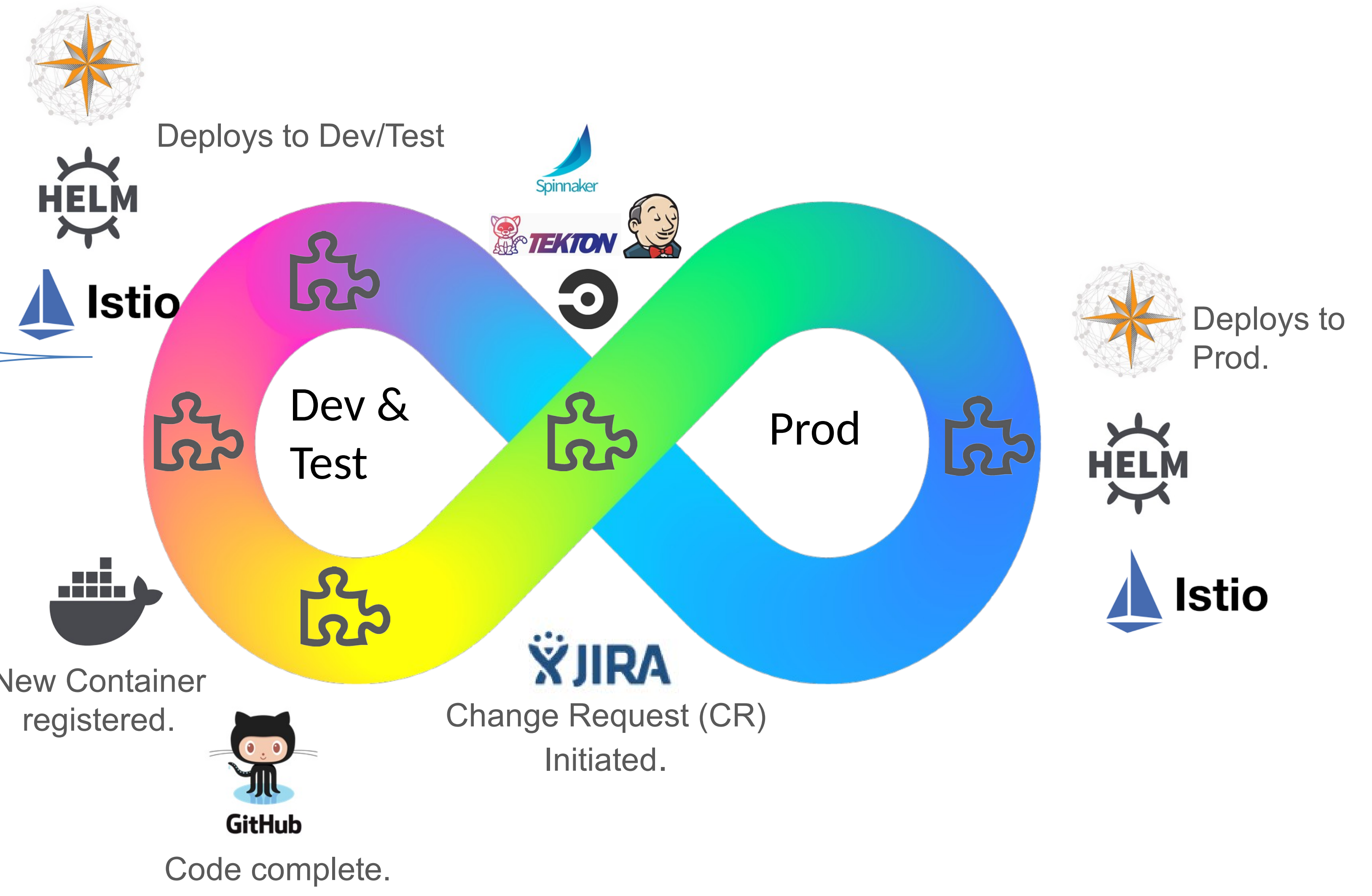


Your New Kubernetes CD Pipeline

Provides BOM, Difference and Impact Analysis Reports.

Establishes Logical Application versions.

Pulls SHA, Git Commit and CR. Creates a new microservice version for sharing based on Domains.



Thank you

LinkedIn: https://www.linkedin.com/in/steve-taylor-oms/
Twitter: @SBTaylor15
Calendar: https://drift.me/stevetaylor/meeting/coffeechat
Email: Steve@DeployHub.com

Dig In at: DeployHub.com or Ortelius.io

DEVOPS WORLD by CloudBees