Zimbra™ Infrastructure Orchestration

continuous availability and scalability through automation

Jared Reimer, CTO
Cascadeo Corporation
jared@cascadeo.com

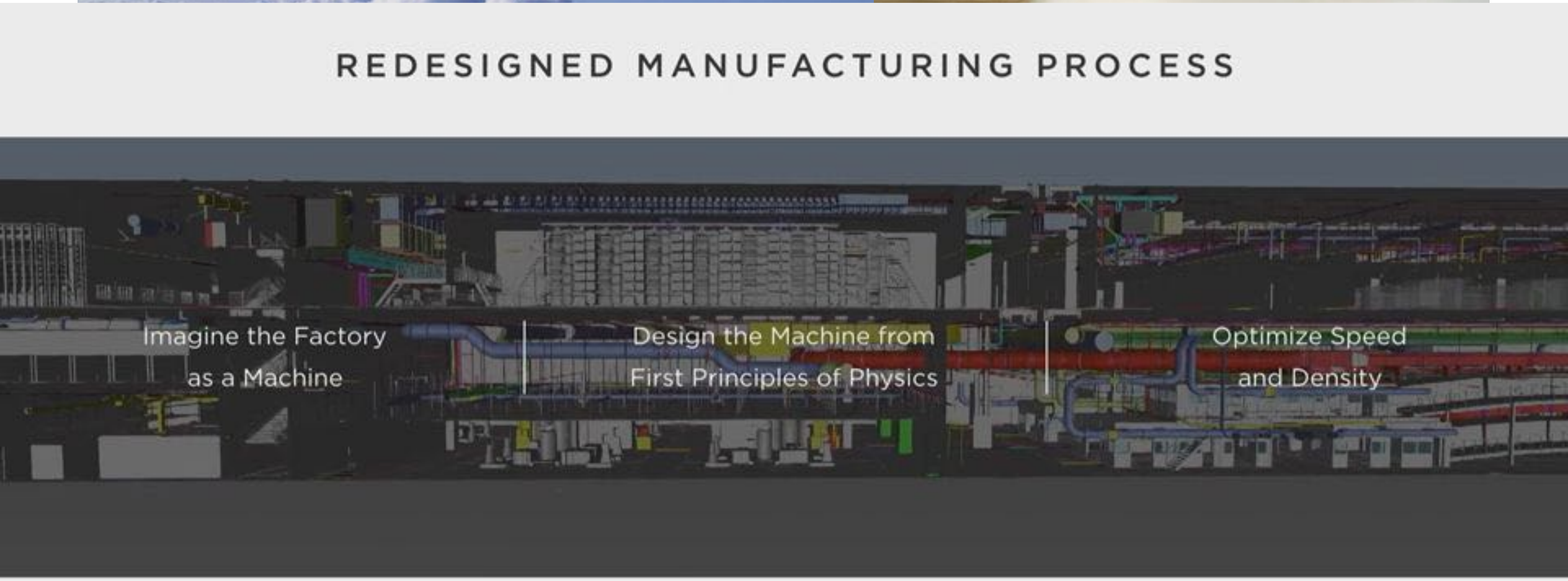
Quick Agenda ~30 minutes + Q&A

- ❑ Intro / Bio; Asks & Promises
- ❑ A Brief History of High-Availability Systems
- ❑ The End of the Runway (*is always closer than it looks*)
- ❑ Compromise Thoughtfully
- ❑ A Solution Architecture for Zimbra™ Orchestration
- ❑ Q&A / Open Discussion

100% of All Infrastructure Fails Eventually

Firefighting in Production

The A380 and the Tesla GigaFactory


REDESIGNED MANUFACTURING PROCESS

Imagine the Factory as a Machine

Design the Machine from First Principles of Physics

Optimize Speed and Density

Tesla's approach to the design of the Gigafactory

A Brief History of High-Availability Strategies

- Active:Standby / failover / data replication
- Active:Standby:Witness / quorum
- Active:Active / load-sharing
- Active:Active:Active / stateless server infrastructure
- N-active + passive failover / GTM

Why Even 3x Redundancy Isn't Always Enough

- **Black Swan events happen every single day**
  - Cascading failures due to tight coupling
  - Shared storage and network assets
  - DNS, Humans, and many other SPOFs that won't go away
- **Complex systems will fail in unpredictable ways**
  - Partial failures often worse than total failure
  - Some bugs occur rarely and are hard to repro in lab
- **Humans are often our own worst enemies**
  - "The road to hell is paved with good intentions"

The End of the Runway for Conventional Ops

- **Legacy cruft piles up by iterative patching**
  - Unknown security and ops history; entropy is the enemy
  - Total inability to detect modern rootkits / malware
- **No repeatability = no recoverability**
  - Documentation is always wrong when you need it most
  - Most DR/BC systems rarely exercised in production
  - One-offs and customizations are incompatible with SaaS
- **Band-aids always fossilize into permanent fixtures**
  - Focus rarely returns after the fire-drill ends
  - Refactoring hand-built IT: the job no dev wants

No Easy Answers?

- **Availability, Durability, Scalability, $ goals at odds**
  - Many Zimbra user populations with *very* different needs
- **No single "correct answer" to the core problems**
  - Lower-level infrastructure redundancy (cost)
  - Higher-level data & application redundancy (complexity)
  - Automation is not free and sometimes overkill
- **Existing deployments and infrastructure have value**
  - Preserve value in existing investments and talent pool
  - Provide a clear migration path to public / private cloud
  - Don't force the hand of the customer or impose platform

Make Thoughtful Compromises to Scale

AVAILABILITY

Understand what "**good enough**" is.

Endpoints responsive in all failure scenarios.

DURABILITY

What is your worst case tolerance for data loss?

Loss != temporary inability to access data.

SCALABILITY

Spread traffic across parallel production stacks.

Shard the workload; rebalance continuously

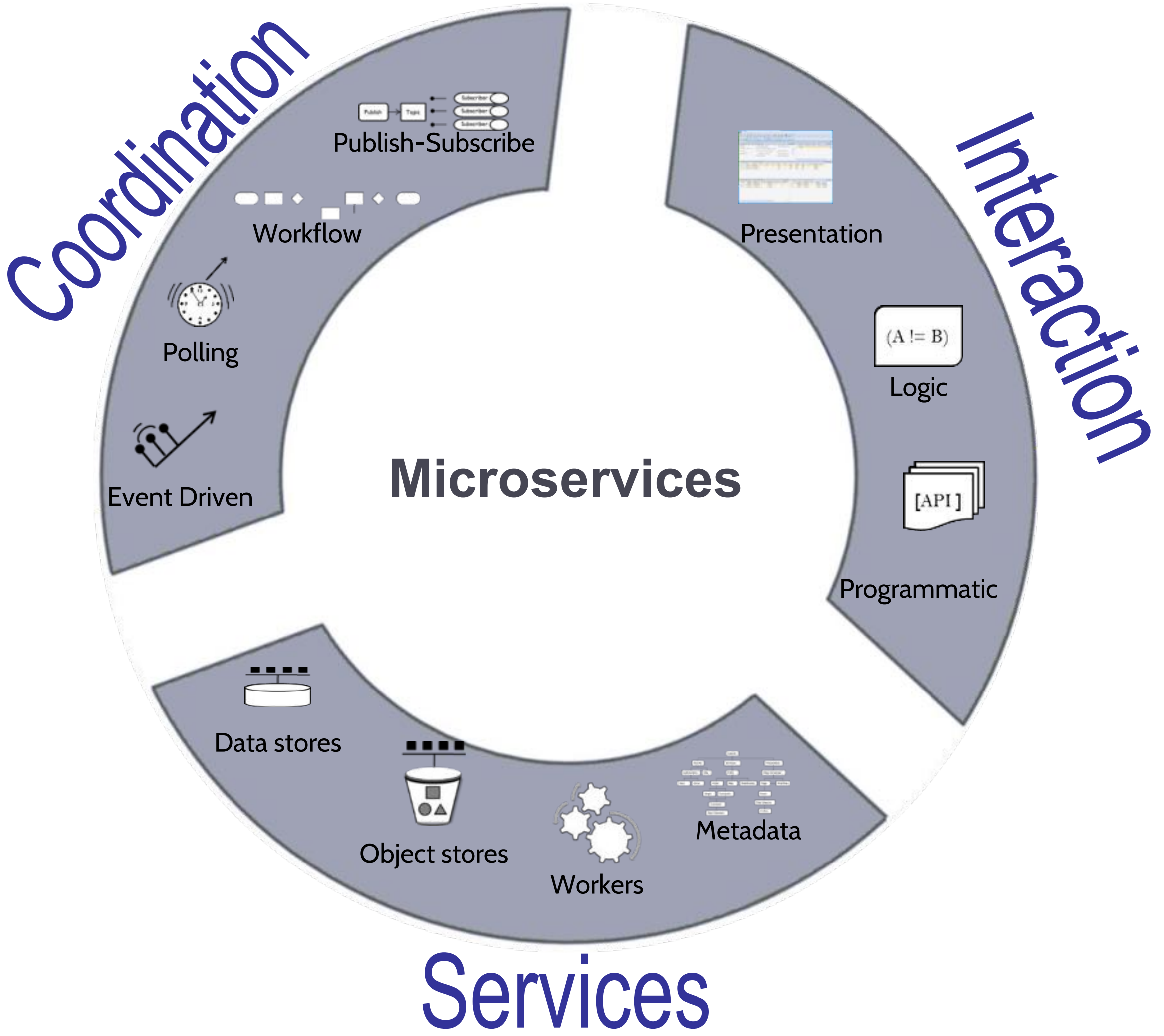
SECURITY

Incidents inevitable.

Aim for limited blast radius.

Limit human access.

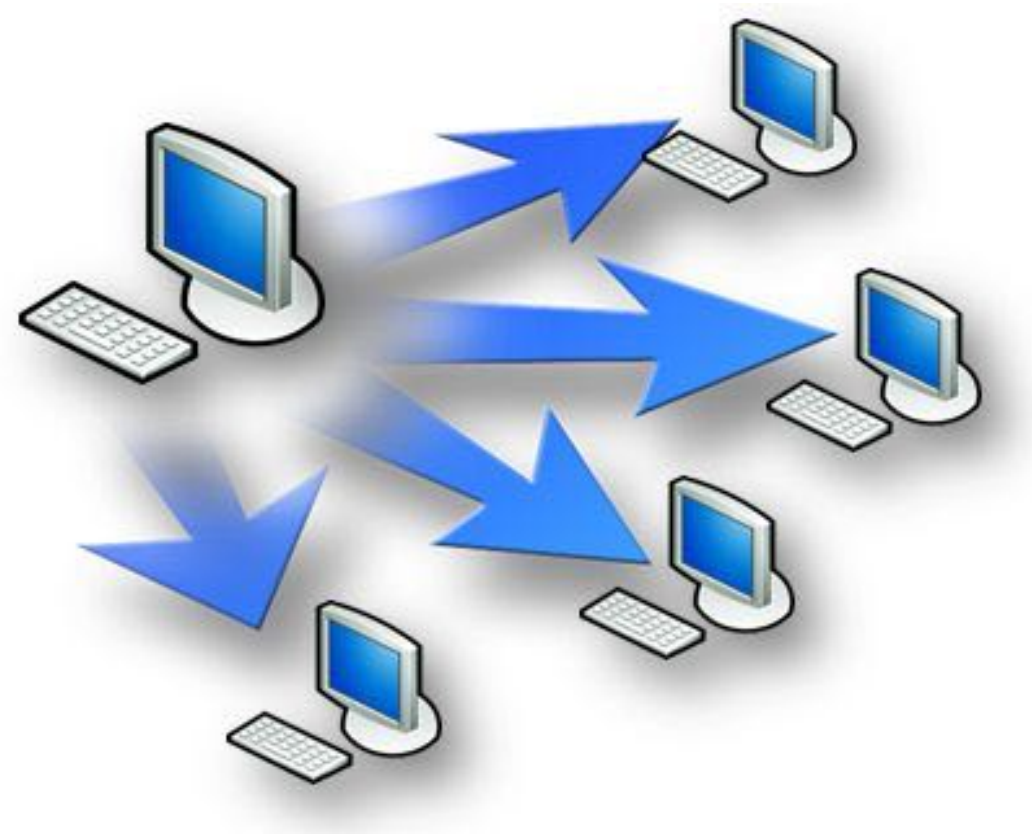
Service Oriented Computing using Microservices



A **solution** is a collection of **microservices** split into three primary categories:

Coordination, Interaction and **Services**

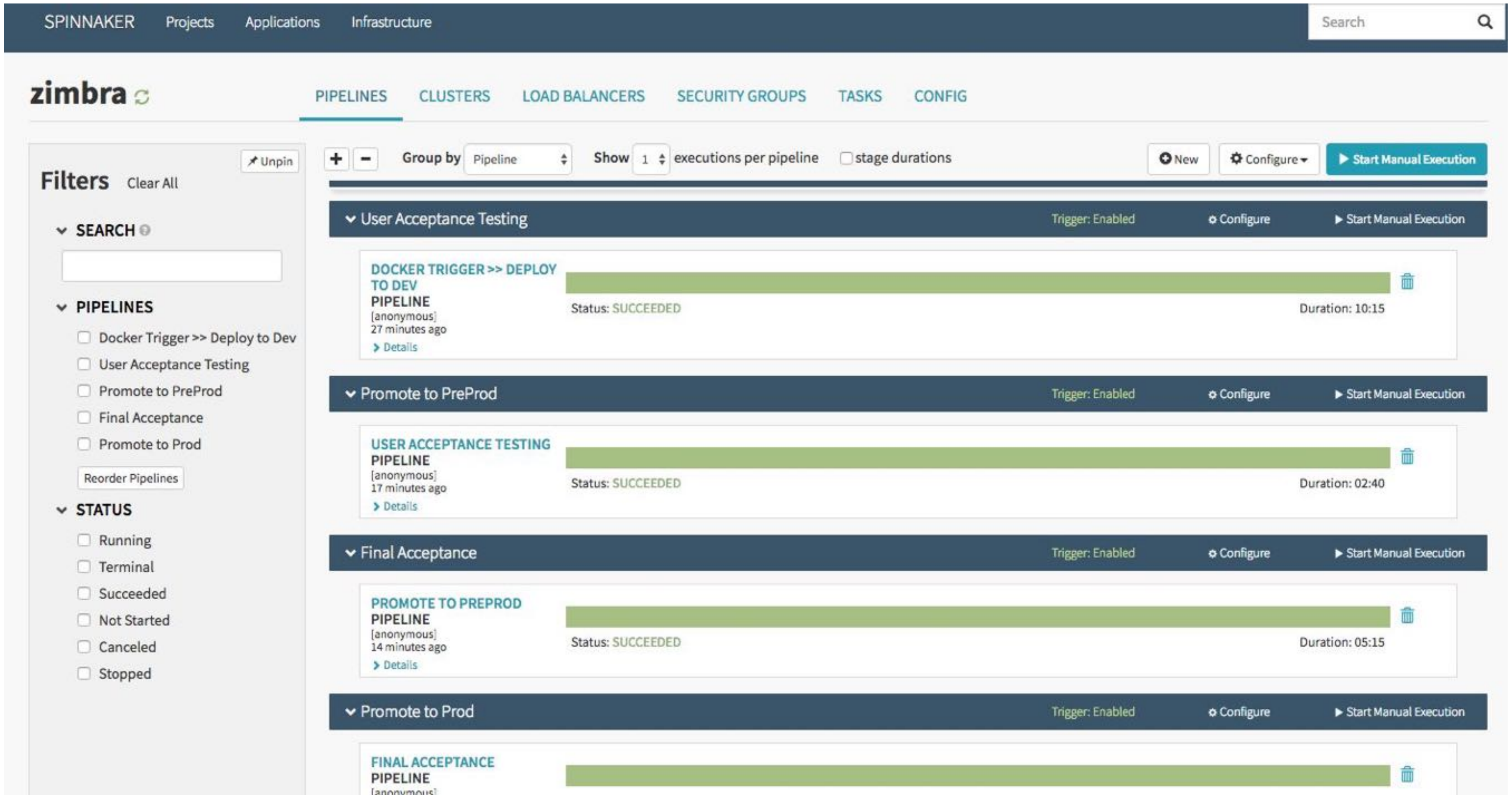
Immutable Deployments



Changes to production systems are always delivered through controlled **replacement** operations.

Continuous Deployment w/ Spinnaker

Spinnaker provides automated pipelines for continuous delivery.



SPINNAKER Projects Applications Infrastructure Search

zimbra

PIPELINES CLUSTERS LOAD BALANCERS SECURITY GROUPS TASKS CONFIG

Filters Clear All Unpin

SEARCH

PIPELINES
- Docker Trigger >> Deploy to Dev
- User Acceptance Testing
- Promote to PreProd
- Final Acceptance
- Promote to Prod

Reorder Pipelines

STATUS
- Running
- Terminal
- Succeeded
- Not Started
- Canceled
- Stopped

Group by Pipeline Show 1 executions per pipeline stage durations New Configure Start Manual Execution

User Acceptance Testing — Trigger: Enabled — Configure — Start Manual Execution

DOCKER TRIGGER >> DEPLOY TO DEV PIPELINE [anonymous] 27 minutes ago Details — Status: SUCCEEDED — Duration: 10:15

Promote to PreProd — Trigger: Enabled — Configure — Start Manual Execution

USER ACCEPTANCE TESTING PIPELINE [anonymous] 17 minutes ago Details — Status: SUCCEEDED — Duration: 02:40

Final Acceptance — Trigger: Enabled — Configure — Start Manual Execution

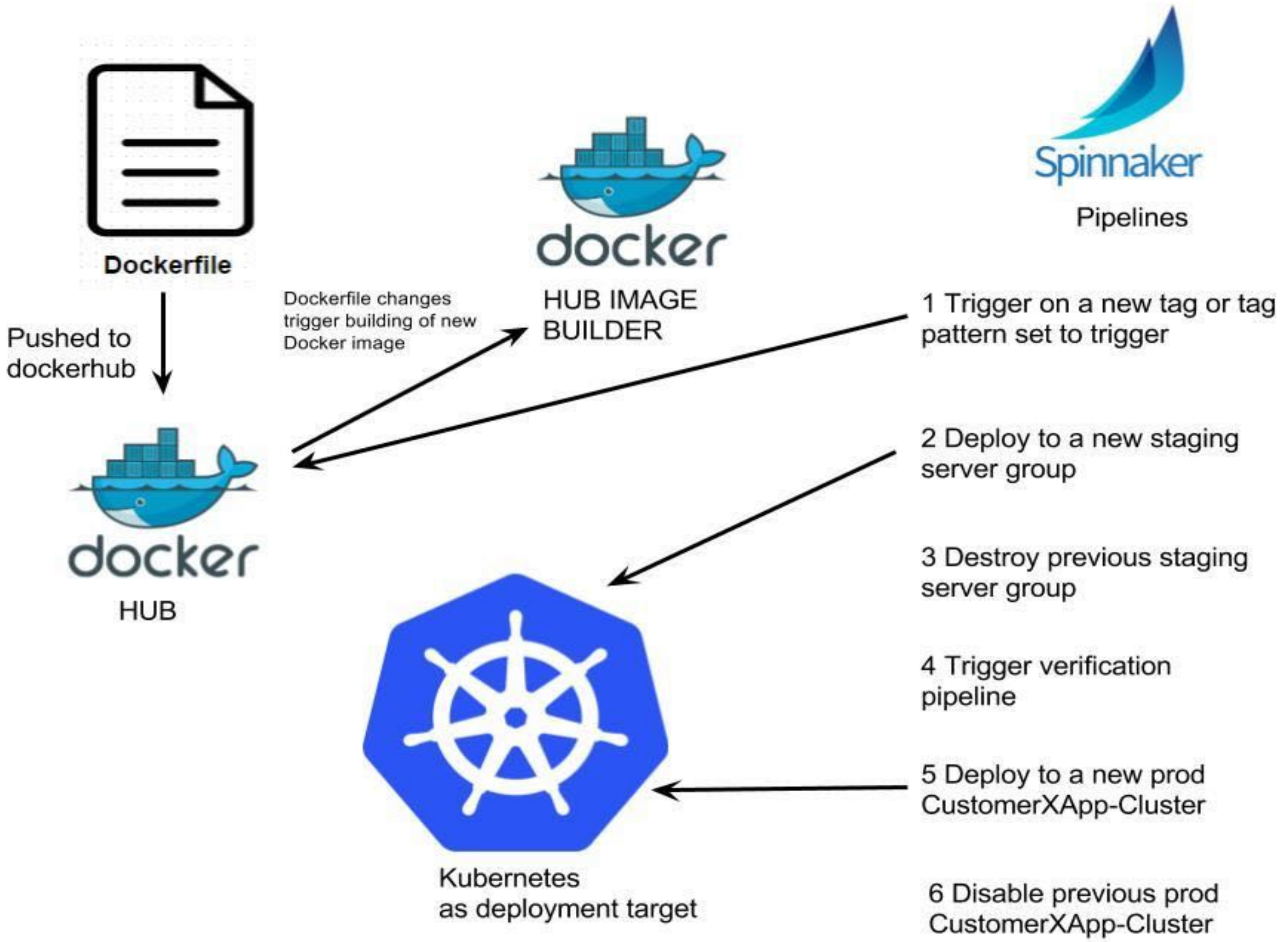
PROMOTE TO PREPROD PIPELINE [anonymous] 14 minutes ago Details — Status: SUCCEEDED — Duration: 05:15

Promote to Prod — Trigger: Enabled — Configure — Start Manual Execution

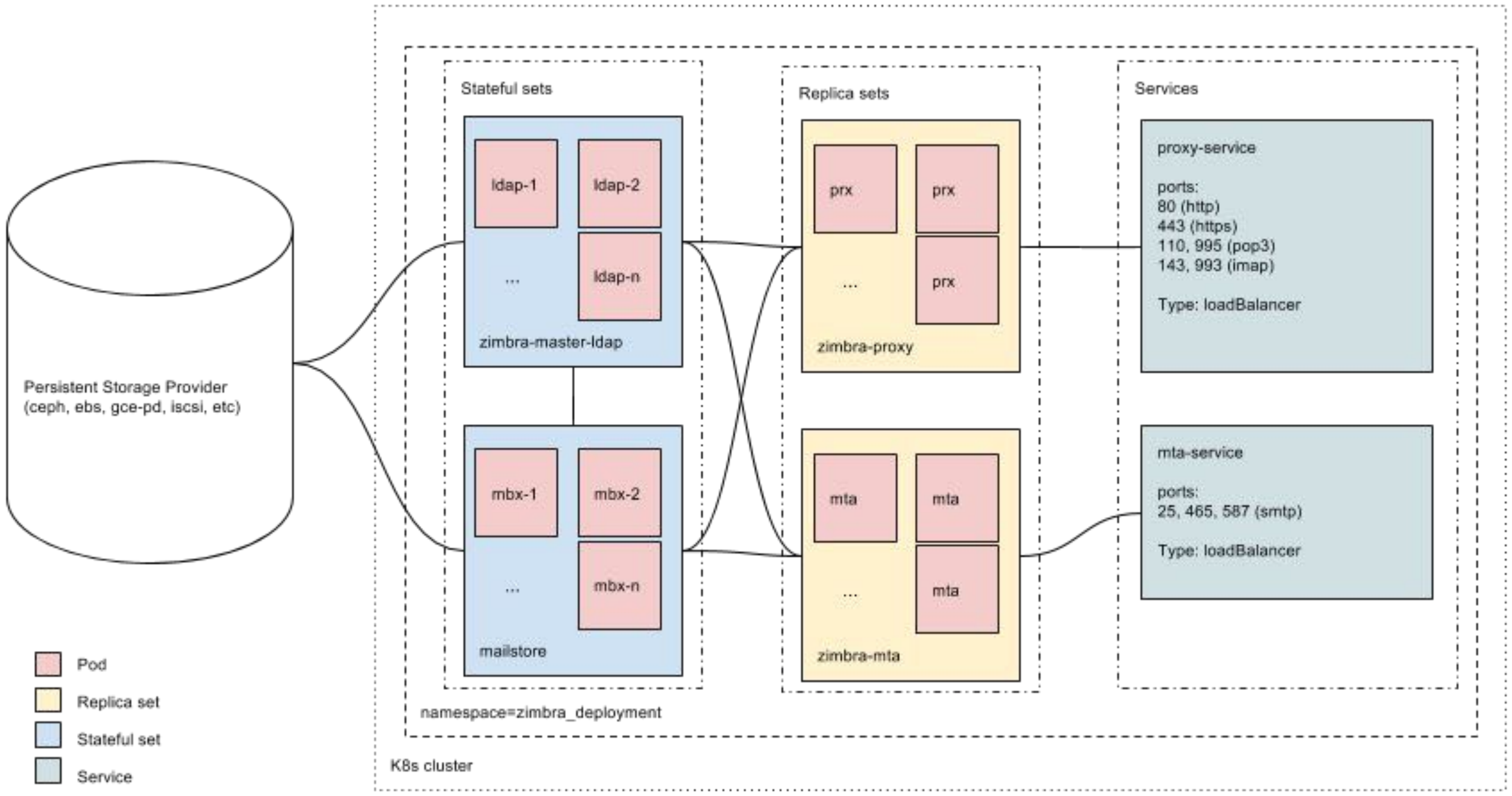
FINAL ACCEPTANCE PIPELINE

cascadeo

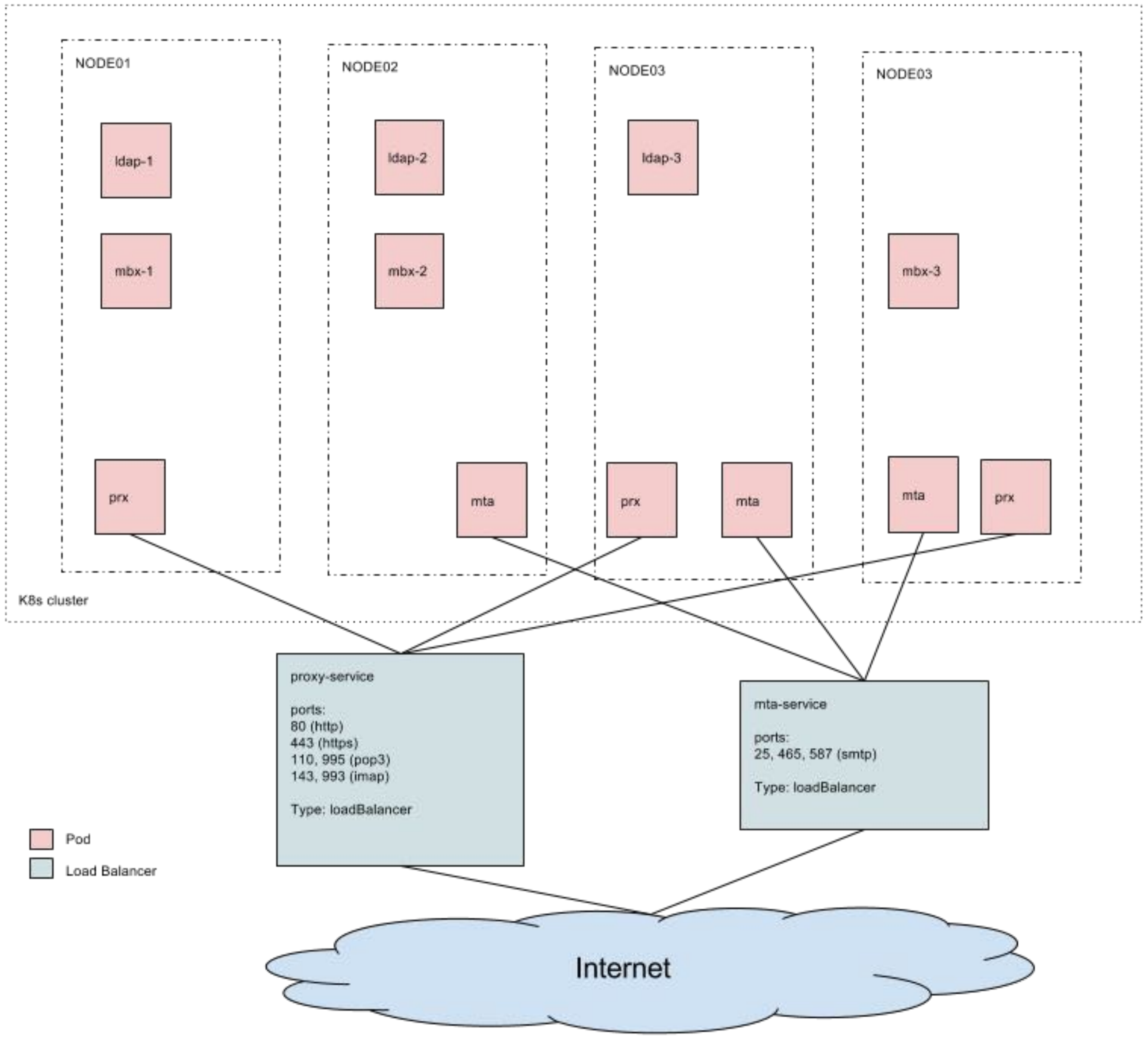
CD Pipeline for Zimbra™ Server Groups


Dockerfile
Pushed to dockerhub
docker
HUB
Dockerfile changes trigger building of new Docker image
docker
HUB IMAGE BUILDER
Spinnaker
Pipelines
1 Trigger on a new tag or tag pattern set to trigger
2 Deploy to a new staging server group
3 Destroy previous staging server group
4 Trigger verification pipeline
5 Deploy to a new prod CustomerXApp-Cluster
6 Disable previous prod CustomerXApp-Cluster
Kubernetes as deployment target

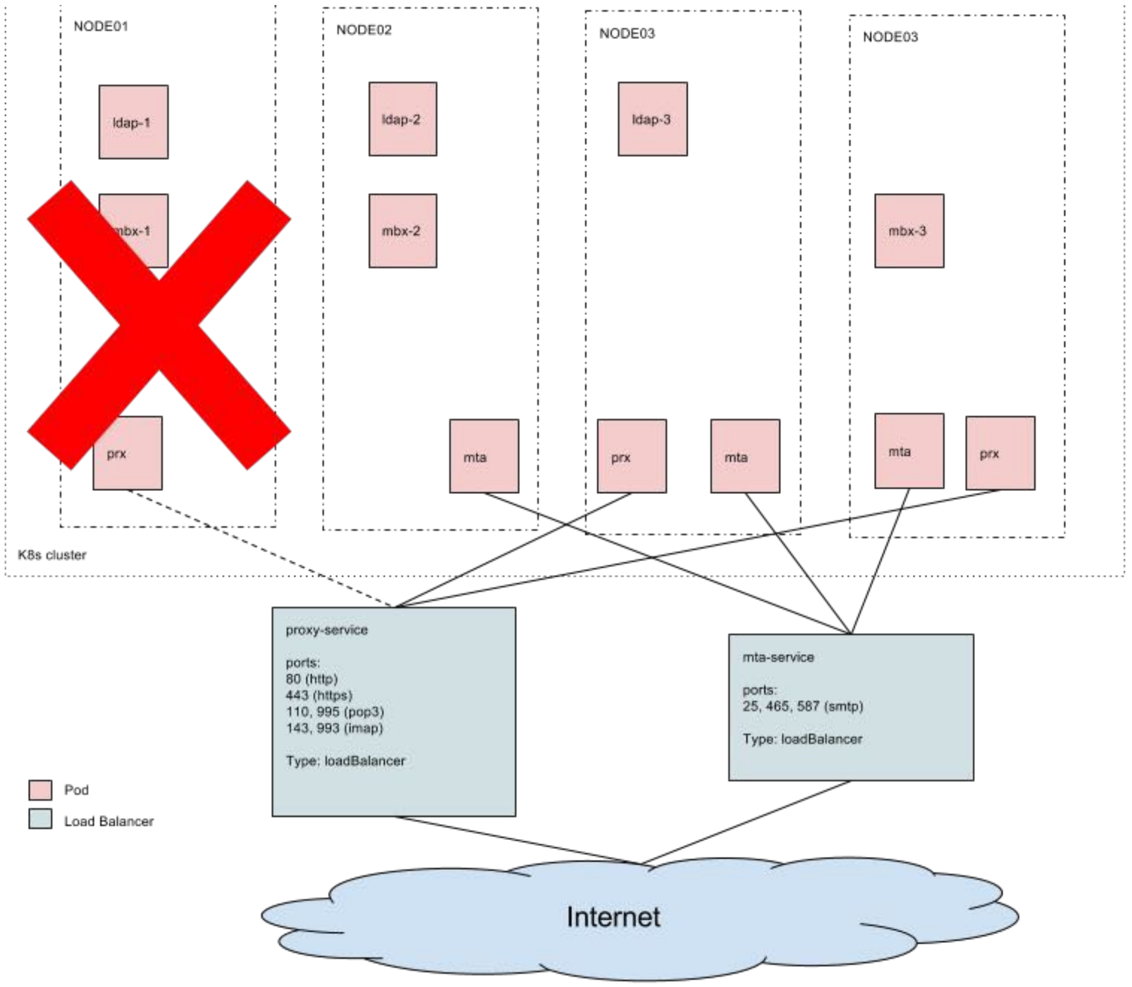
Scaling Stateful and Stateless Microservices



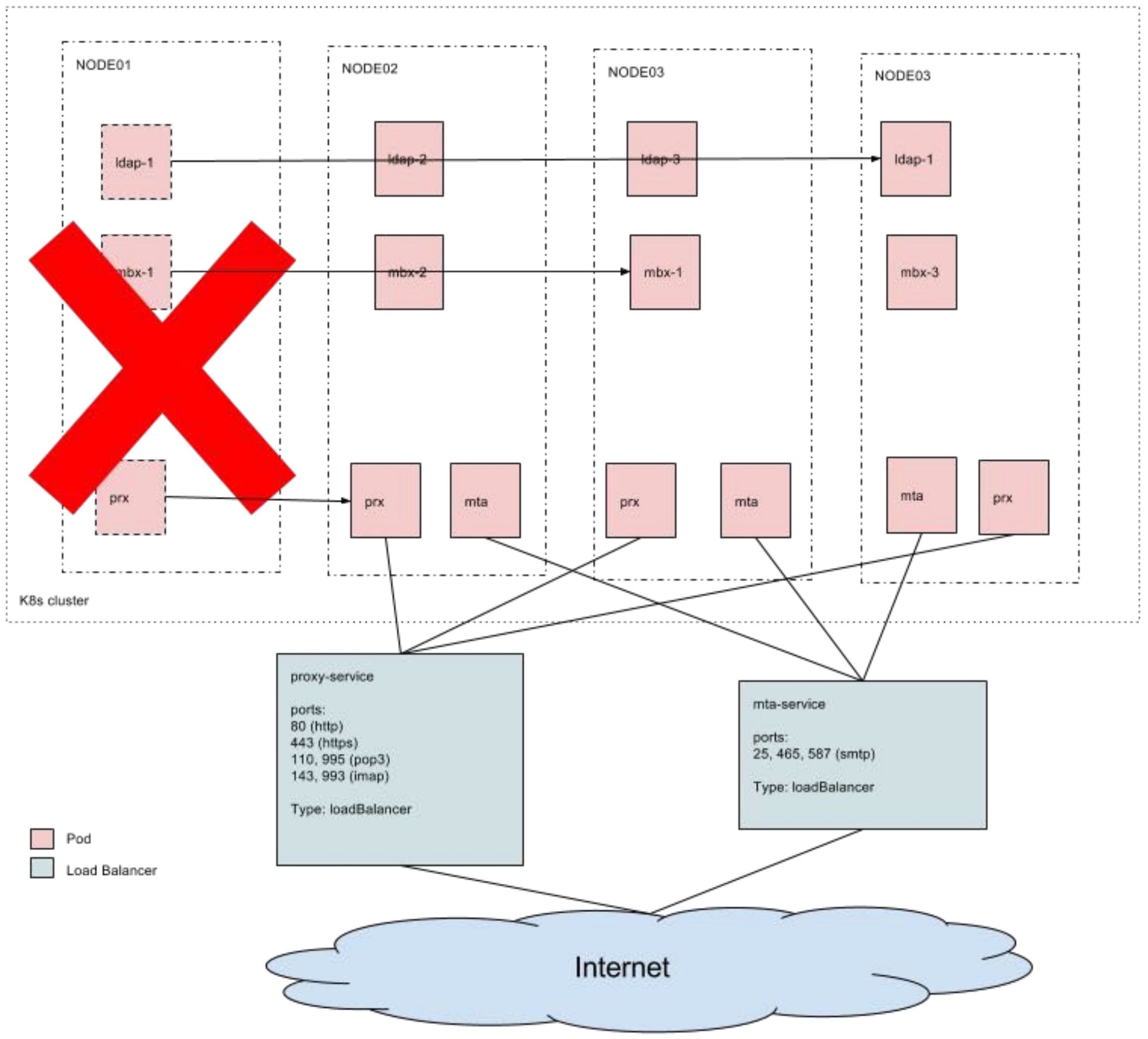
Example: Four-Node Kubernetes cluster



Spontaneous / Uncontrolled Node Failure



Automated Stabilization and Remediation



Continuous Stress Testing

Prove, **every single day**, that our self-healing capability is functioning as expected... (*Thanks, Netflix!*)

Faults that repeatedly fail to clear themselves are likely bugs.

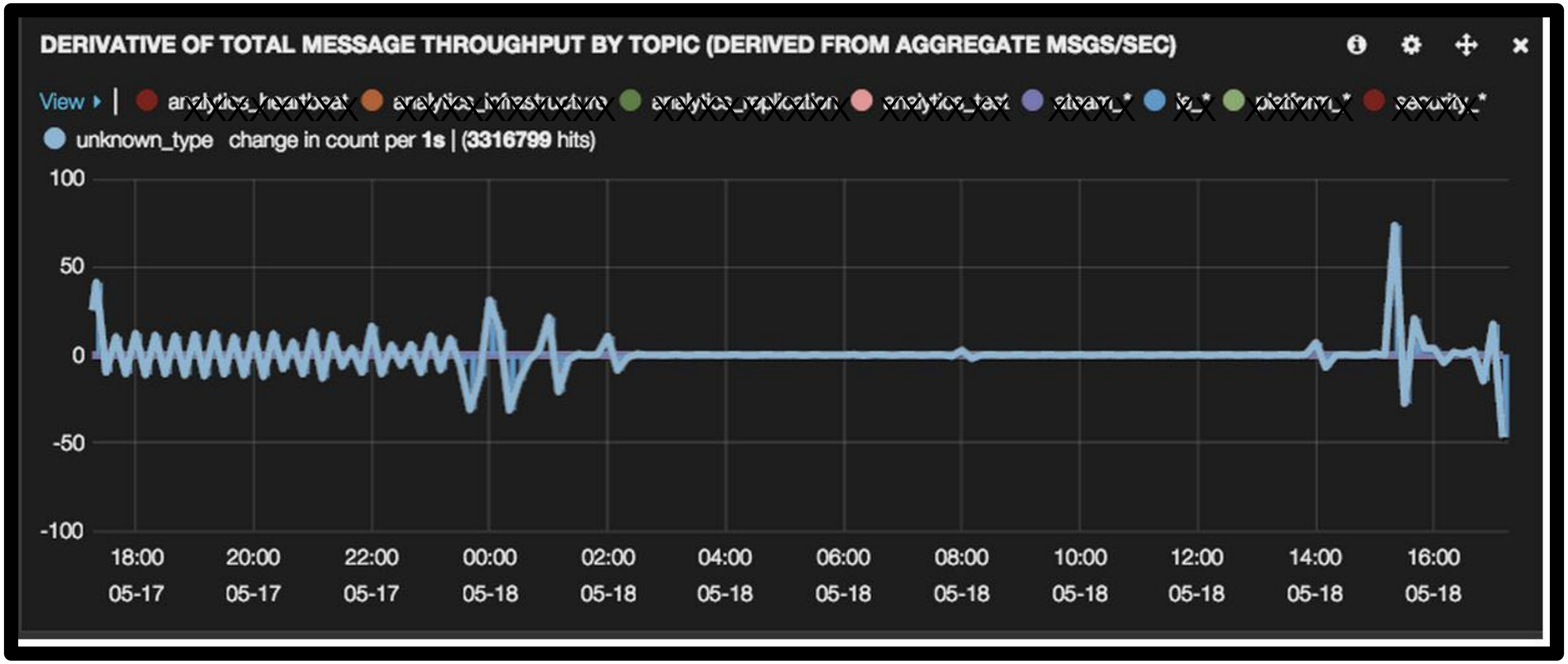
DR/BC strategy is exercised continuously, in production.

Predictive Analytics & Anomaly Detection

Machine learning algorithms detect subtle changes and relationships between a wide range of signals.

Predict and repair faults -- often before they become impactful.

Disturbances in the Force



- a picture is worth a thousand words
- if a derivative falls in the woods and nobody hears it...

Summary and Key Conclusions

- **Continuous Delivery and Stress Testing**
  - Deploy / replace rather than “fixing things” by hand
  - Repeated faults are bugs, not ops incidents.
- **No Repeatability = No Recoverability**
  - Configuration treated as code and deployed accordingly
  - Band-aids are quickly replaced by automation
- **Understand the Worst Case and Work Backwards**
  - How much is “good enough”?
  - What is an acceptable brown-out or loss window?
  - Limit the blast radius and preserve the user experience!

cascadeo

Open Discussion / Q&A / Thanks!

email welcomed jared@cascadeo.com
or iMessage / SMS +1 (206) 650-7090