Evolving DevOps to GitOps

A Perspective

Bob Wise
GM, Kubernetes
wisebob@amazon.com
@countspongebob
Motivators

Leadership knows velocity is critical to competitiveness.
Massive velocity gap between high vs average orgs.
Desire to constrain sprawl, use fewer, better tools and approaches.
Teams needing to modernize for retention and recruiting.

*Pressure to evolve*...
Evolution in Processes

Spiral to Slowness

- Team discord
- Larger release
- Start manual sign-offs
- Zero-failure policy implemented
- Lengthen soak test
- Exec sign-off before release
- Impact of failure much wider
- Manual tests
- More tests
- Failed use reports
- Re-Install
- RCA—shame
- Regression
- Auto Tests
- Manual/Auto test
- Release
- Auto Rollback
- Fix
- Redeploy
- Smooth Rollback
- Monitor (publicly)

Great team

- More Manual tests
- Only roll-forward possible
- Make Stage a full replica
- Test cycle time increases
- Release gets bigger
- Impact of failure much wider
- Zero-failure policy implemented
- Small group of superstars willing to risk change/install
- RCA—shame
- Try Again
- Exec sign-off before release
- Impact of failure much wider
- Fix
- Redeploy
- Smooth Rollback
- Monitor (publicly)

High Velocity

- Only roll-forward possible
- Make Stage a full replica
- Test cycle time increases
- Release gets bigger
- Impact of failure much wider
- Zero-failure policy implemented
- Small group of superstars willing to risk change/install
- RCA—shame
- Try Again
- Exec sign-off before release
- Impact of failure much wider
- Fix
- Redeploy
- Smooth Rollback
- Monitor (publicly)

Velocity Flywheel

© 2019, Amazon Web Services, Inc. or its Affiliates.
Effective Strategies

Focus on modern operations (CD) as the chokepoint to velocity over development improvements (Agile dev and CI).

Use container orchestration to realize CI and CD.

Incrementally evolve the entire org, not just Greenfield.

Treat operations hygiene as the critical prelude to microservice re-architecture: CD + observability + orchestration
Evolution of Approaches

People
- Package managers
- Config Databases
- Siloed Teams/Tools
- Focus on Dev Productivity

Robots
- Immutable images
- Configuration as Code
- Integrated Teams/Tools
- Holistic Velocity
Evolution Of Approaches

- Manual Processes
- Package managers
  ...and Bash! 😊
- Spreadsheets 😊
- Dev->QA->Security->Ops
- Focus on CI

- Automation
- Containers
- Configuration as Code
- DevOps
- Focus on CD
Fully Automated Deployment is HARD.
Deployment Challenges

- Complicated imperative dance
- Ordered steps
- Dependency tracking
- Packaging
- Versioning
- Rollbacks
Evolution of Ops and CD Approaches

- Imperative Automation → Declarative Automation
- Automated Mutation → Immutability
- Single State → Reconciliation/Convergence
Evolution of Ops and CD Approaches

DevOps ➔ GitOps
GitOps

Everything in source control

Declarative configuration

Robots do all the work

Continuously converged and healed
Everything in Source Control

Managed with software engineering practices.

Human readable source of truth.

Statement of intent by the humans.

Reviewed before deployment by best practice and org policy.

Deployments triggered by merging the PR.
Declarative

Easier to reason about at scale.

Disciplined simplification.

Define intent, let robots do the work to implement.

Convergence point.
Robots aren’t our overlords yet: They do all the work

Most orgs terrified if humans aren’t doing deployments.

High velocity orgs are terrified if they are.
Continuous Convergence / Drift Management

Diagram:
- Desired State
- Act
- Measure
- Compare

© 2019, Amazon Web Services, Inc. or its Affiliates.
What’s Next?

CD  ➔  PD
What’s Next?

Continuous Deployment → Progressive Deployment
What’s Next?

Continuous Deployment → Progressive Deployment

Why?
Hypocenter

Key:
Type of destruction
(radius in miles/kilometers)

Complete destruction
(1 mi / 1.6 km)

Severe damage
(1.25 mi / 2 km)

Moderate damage
(2 mi / 3.2 km)

Light damage
(3.5 mi / 5.6 km)
Progressive Deployment

Roll out slowly, and if all tests are still passing, accelerate the pace of rollout exponentially, otherwise roll back automatically. Target 24 hours for full deploy.

Roll out slowly but fully into one region, once it has baked for a week, roll out every other region rapidly.

Roll out quickly to user profile X first, then to profile Y…
Progressive Deployment

Roll out slowly, and if all tests are still passing, accelerate the pace of rollout exponentially, otherwise roll back automatically. Target 24 hours for full deploy.

Roll out slowly but fully into one region, once it has baked for a week, roll out every other region rapidly.

Roll out quickly to user profile X first, then to profile Y…

- Blast radius management
- Fine control over deployment strategy and execution
- Control rollout scope
- Who sees it and how much of it gets out where
Fully Automated Deployment is Complicated…

*But the robots are here to help...*
Kubernetes: Made for GitOps

- Supports declarative application management
- Idempotency
- Convergence
  - Controllers
  - Operators
- Immutability
- Deployments
- Ingress
- Organic healing properties
Argo CD and Flux CD are joining forces!

- Best of Flux CD and Argo CD
- Open source collaboration
- Expands GitOps ecosystem

Join us in evolving Argo CD and Flux CD!

https://github.com/argoproj/gitops-engine/