DevSecOps

IT modernization through DevSecOps
Tyler Johnson

Federal Solutions Architect CloudBees
Tyler is a Solutions Architect at CloudBees focusing on the needs of Federal sector agencies. With a professional services background Tyler works closely with prospects to drive business value through technical solutions. He specializes in helping organizations move toward DevSecOps while establishing structured governance and standardization that large enterprises demand.

Contact
tjohnson@cloudbees.com
https://www.linkedin.com/in/tyler-johnson-rva/
Software delivery organizations encounter risk and inefficiency from disconnected groups, teams, and tools

Disconnected development and operations frequently breeds conflict and inefficiency, exacerbated by competing motivations and processes. - Wired

84% of respondents say inaccessibility of information gets in the way of their ability to do their jobs and/or make data-driven decisions. Respondents attributed this to organizational and functional silos. - ASG 2020 State of Software Delivery

59% of organizations cannot keep an audit trail of who changed what and when - DevSecOps 2019

Lack of integrated, end-to-end development, delivery and release

Lack of end-to-end visibility into the SDLC

Lack of traceability and governance
Connect Fragmented Software Delivery Lifecycle across Groups, Teams and Tools

Software Delivery in Silos

- Each function uses disconnected point tools, processes and data.
- Inconsistencies across groups and teams creates friction and slows pace
- Errors, rework and redundancies escalate cost and audit risk

CloudBees Software Delivery Automation

- Formal enterprise scale connected CI/CD platform architecture provides governance and reuse.
- Entities process and assets readily available across CI/CD
- Shared orchestration across all groups, teams and tools provides full visibility across the SDLC
What is the Software Supply Chain?

Traditional supply chain:
- Gear
- Factory
- Truck
- Store
- Customer

Software supply chain:
- Source/dependencies
- Build systems/engines
- Network
- Application repository
- Deployed systems
Increasing confidence in build's production readiness

Environments become more production like

Faster feedback

Basic Deployment Pipeline
Increasing confidence in build’s production readiness

Define your best practices - Choose your own tools

Developer:
- based on user story(s) assigned, new request to IT catalog
- get request approval and artifacts
- updates code and tests locally
- checks in code, tests and configuration

Commit Stage
- Compile/Build
- Unit Test
- Assemble
- Code Analysis

Acceptance Stage
- Configure Environments
- Deploy Binaries
- Smoke Tests & Acceptance Tests
- Testers
- Self-service deployments

Release Orchestration
- Configure Environments
- Deploy Artifacts
- Smoke Test
- CD Pipeline

Production Feature Deployment
- Users progressively exposed to new features in production
- Feature Flags
- Stored in Application Code Base

Product Owners
- Determine release approach

Commit
- Build
- Scan & Test

Build/Configure Environment
- Configure Environment
- Deploy Binaries
- Smoke Tests & Acceptance Tests

Testers
- Self-service deployments

Releases
- Configure Environments
- Deploy Artifacts
- Smoke Test
- ATO Approval

Reports
- Binaries
- Metadata

Users
- Post Commit Stage: Canary & BG Deployments w/ A/B Testing
- Post Acceptance Stage: Early Adopters
- Full Acceptance: Deployment of Features for all users

Artifact Repository
- Sonatype
- JFrog Artifactory
- Eclipse Nexus
- JFrog Stash

CloudBees
- Orchestration
- Pre-Production
- Production

Acceptance Stage
- Configure Environments
- Deploy Artifacts
- Smoke Tests & Acceptance Tests

Faster feedback

Developer:
- based on user story(s) assigned, new request to IT catalog
- get request approval and artifacts
- updates code and tests locally
- checks in code, tests and configuration

Commit Stage
- Compile/Build
- Unit Test
- Assemble
- Code Analysis

Acceptance Stage
- Configure Environments
- Deploy Binaries
- Smoke Tests & Acceptance Tests
- Testers
- Self-service deployments

Release Orchestration
- Configure Environments
- Deploy Artifacts
- Smoke Test
- CD Pipeline

Production Feature Deployment
- Users progressively exposed to new features in production
- Feature Flags
- Stored in Application Code Base

Product Owners
- Determine release approach

Commit
- Build
- Scan & Test

Build/Configure Environment
- Configure Environment
- Deploy Binaries
- Smoke Tests & Acceptance Tests

Testers
- Self-service deployments

Releases
- Configure Environments
- Deploy Artifacts
- Smoke Test
- ATO Approval

Reports
- Binaries
- Metadata

Users
- Post Commit Stage: Canary & BG Deployments w/ A/B Testing
- Post Acceptance Stage: Early Adopters
- Full Acceptance: Deployment of Features for all users

Artifact Repository
- Sonatype
- JFrog Artifactory
- Eclipse Nexus
- JFrog Stash

CloudBees
- Orchestration
- Pre-Production
- Production

Acceptance Stage
- Configure Environments
- Deploy Artifacts
- Smoke Tests & Acceptance Tests

Faster feedback
Modern SW delivery drives mission objectives

Increase Innovation

Drive Innovation Quickly

Enhance software development and delivery to increase speed, time to value, and getting capabilities into end users' hands.

Reduce Risk

Reduce Risk by Ensuring Adherence to Internal and External Policies

Ensure a secure, compliant process and environment that produces safe, reliable outputs.

Increase Efficiency & Reduce Cost

Optimize Software Development and Delivery Function

Eliminate wasted effort to free up staff to focus on mission-critical activities.
Who is DORA?

DORA’s research provides insight into software development and DevOps practices applied in industry, backed by scientific studies spanning six years with over 31,000 survey responses from working professionals. DORA releases the “State of DevOps” report yearly summarizing their findings and setting the bar for DevOps performance.

- Accelerate State of DevOps Report, 2019
Top Performers

ELITE PERFORMERS
Comparing the elite group against the low performers, we find that elite performers have...

208 TIMES MORE frequent code deployments
106 TIMES FASTER lead time from commit to deploy

2,604 TIMES FASTER time to recover from incidents
7 TIMES LOWER change failure rate (changes are 1/7 as likely to fail)