Speaker

Priyanka Sharma
Director of Technical Evangelism - GitLab
Board Member - Cloud Native Computing Foundation
@pritianka
Primer

Continuous Delivery

Unit Test → Platform Test → Deliver to Staging → Application Acceptance tests → Deploy to Production → Post deploy tests

Continuous Deployment

Unit Test → Platform Test → Deliver to Staging → Application Acceptance tests → Deploy to Production → Post deploy tests

Cloud native development and accelerating cycle time is increasingly important to business survival.

To succeed, engineering leaders must embrace CI/CD. This is GitLab’s story to moving a large scale production system to continuous delivery.

- The business imperative - cycle time compression
- How we did it - legacy first
- The value CD brought to GitLab
- DevOps - achievement unlocked - some advice
Business survival depends upon a radically faster DevOps lifecycle which can only be enabled by shifting left.
Speeding Up Release Cycle Time is Critical to Business

Cycle time compression may be the most underestimated force in determining winners & losers in tech.

— Marc Andreessen
How: Small, very small changes unlock velocity
Cloud native development and accelerating cycle time is increasingly important to business survival.

To succeed, engineering leaders must embrace CI/CD. This is GitLab’s story to moving a large scale production system to continuous delivery.

- The business imperative - cycle time compression
- How we did it - legacy first
- The value CD brought to GitLab
- DevOps - achievement unlocked - some advice
GitLab runs on GitLab, uses Ansible, and does not yet use k8s for CD.
Ok we do use a little bit of Kubernetes and feature flags

**All engineers** need access to environments (infrastructure) on demand for testing, evaluation and gradual rollout (i.e. canary deploys, etc)
Ability to review changes live - this is based on k8s
Cloud native development and accelerating cycle time is increasingly important to business survival.

To succeed, engineering leaders must embrace CI/CD. This is GitLab’s story to moving a large scale production system to continuous delivery.

● The business imperative - cycle time compression
● How we did it - legacy first
● The value CD brought to GitLab
● DevOps - achievement unlocked - some advice
GitLab

From weekly to daily deploys

CREATE - PLAN - PACKAGE - RELEASE

DEV - OPS

CONFIGURE - MONITOR

SECURE - DEFEND
GitLab

All developers in on call rotation within 3 weeks
Cloud native development and accelerating cycle time is increasingly important to business survival.

To succeed, engineering leaders must embrace CI/CD. This is GitLab’s story to moving a large scale production system to continuous delivery.

- The business imperative - cycle time compression
- How we did it - legacy first
- The value CD brought to GitLab
- DevOps - achievement unlocked - some advice
Our culture shift

Quality is a priority

No hot-patching unless p1 and s1

Every engineer is on the on-call rotation
All aboard the k8s boat!
GitLab is powered by GitLab

**VISIBLE**
Real time view across the entire lifecycle

**EFFICIENT**
Collaborate without waiting

**GOVERNED**
Develop and operate with confidence
Our humble advice

Figure out the workflows, the rest will follow

Don’t feel compelled to use shiny objects

Decoupling infra stack changes from people workflow changes makes CD much more doable
Thank you!

Priyanka Sharma
Director of Technical Evangelism - GitLab
Board Member - Cloud Native Computing Foundation
@pritianka
GitLab is the first single application for the entire DevOps lifecycle

<table>
<thead>
<tr>
<th>Manage</th>
<th>Plan</th>
<th>Create</th>
<th>Verify</th>
<th>Package</th>
<th>Secure</th>
<th>Release</th>
<th>Configure</th>
<th>Monitor</th>
<th>Defend</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cycle Analytics</td>
<td>Kanban Boards</td>
<td>Source Code Management</td>
<td>Continuous Integration (CI)</td>
<td>Container Registry</td>
<td>Continuous Delivery (CD)</td>
<td>Auto DevOps</td>
<td>Metrics</td>
<td>Runtime Application Self Protection</td>
<td></td>
</tr>
<tr>
<td>DevOps Score</td>
<td>Project Management</td>
<td>Code Review</td>
<td>Code Quality</td>
<td>Maven Registry</td>
<td>DAST</td>
<td>Kubernetes Configuration</td>
<td>Logging</td>
<td>Web Application Firewall</td>
<td></td>
</tr>
<tr>
<td>Audit Management</td>
<td>Agile Portfolio Management</td>
<td>Wiki</td>
<td>Performance Testing</td>
<td>NPM Registry</td>
<td>Dependency Scanning</td>
<td>ChatOps</td>
<td>Cluster Monitoring</td>
<td>Threat Detection</td>
<td></td>
</tr>
<tr>
<td>Authentication and Authorization</td>
<td>Service Desk</td>
<td>Snippets</td>
<td>System Testing</td>
<td>Container Scanning</td>
<td>License Management</td>
<td>Serverless</td>
<td>Tracing</td>
<td>Behavior Analytics</td>
<td></td>
</tr>
<tr>
<td>Coming soon:</td>
<td>Coming soon:</td>
<td>Web IDE</td>
<td>Rubygem Registry</td>
<td>License Management</td>
<td>Incremental Rollout</td>
<td>Coming soon:</td>
<td>Error Tracking</td>
<td>Vulnerability Management</td>
<td></td>
</tr>
<tr>
<td>Code Analytics</td>
<td>Value Stream Management</td>
<td>Design Management</td>
<td>Usability Testing</td>
<td>Helm Chart Registry</td>
<td>Feature Flags</td>
<td>PaaS</td>
<td>Coming soon:</td>
<td>Data Loss Prevention</td>
<td></td>
</tr>
<tr>
<td>Quality Management</td>
<td>Compatibility Testing</td>
<td>Compatibilty Testing</td>
<td>Compatibility Testing</td>
<td>Dependency Proxy</td>
<td>IAST</td>
<td>Runbook Configuration</td>
<td>Incident Management</td>
<td>Status Page</td>
<td></td>
</tr>
</tbody>
</table>
Ticketmaster migrated from Jenkins to GitLab to speed up build time.

15X faster builds
Paessler AG automated QA tasks down from 1 hour to 30 seconds.

120X

Increased QA efficiency

Public Case Study
SVN was a blocker to adopting DevOps. Axway implemented GitLab and went from yearly to biweekly deployments.

26X faster release cycles

OKAY for now - replace with Goldman Sachs when we can