DEVOPS FOR EXECUTIVES

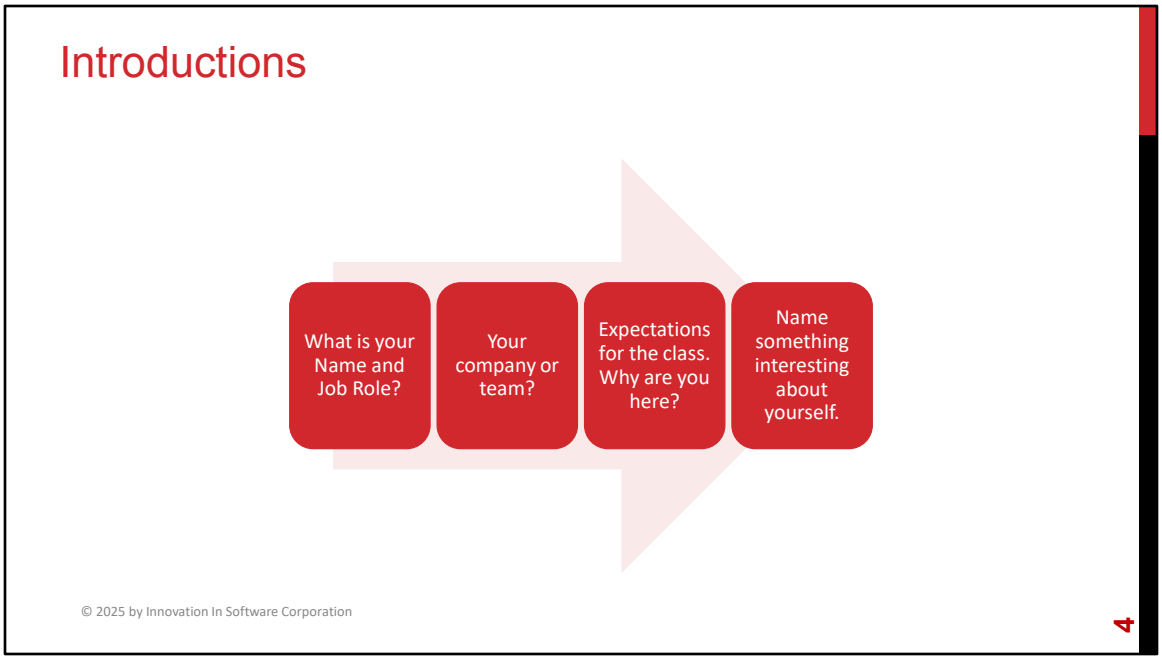
1

TEKsystems Global Services

WORKFORCE DEVELOPMENT

Welcome

- Logistics (breaks, facilities, lunch, etc.)
- Rules of Engagement
- Introductions
- Lets Get Started!



Before diving into the material, it's important to understand who is in the room and what you want to achieve today. This will help me tailor discussions to your organization's needs.

- Name and Job Role: Helps us understand your background and how DevOps fits into your responsibilities.
- Company or Team: Learning about your organization provides insight into possible use cases and challenges.
- Expectations for the Class: Knowing what you're hoping to gain ensures we cover topics most valuable to you.
- Interesting Fact: A light way to connect and build rapport.

Presenter Information

Antoine Victor

MSCE, MCDBA, MCSD, MCT, CSM, CSPO

- Agile Technical Coach, Enterprise IT Engineering Consultant

Workshop Goals and Structure

- Four-Hour Executive Sessions
- Key DevOps Principles for Leaders
- Blend of Practical Insights and Demos

This session is designed with busy executives in mind—concise yet impactful content. Our focus will be on strategic insights and real-world examples.

- Four-Hour Executive Session: The content is streamlined to deliver the highest-value information in the time available, with minimal fluff.
- Key DevOps Principles for Leaders: By exploring frameworks like the Three Ways (Flow, Feedback, Learning), we'll link them directly to measurable organizational outcomes.
- Blend of Practical Insights and Demos: Real-world examples and live demonstrations make the concepts tangible, helping you visualize their application.

What to expect from this workshop

- Flexibility
- Conversations
- Literacy and awareness on the many principles, tools and practices associated with this thing called “DevOps”
- A priority of focus on human behavior first, technology and tools second
- A lot of talk about organizational culture
- An effort to focus on your own situations and challenges so you can act on what you learn

7

This workshop isn’t about rigid rules—it’s about flexibility and conversation. You’ll walk away with insights into how DevOps can help you tackle unique organizational challenges while building a sustainable culture of continuous improvement.
This workshop emphasizes dynamic engagement and real-world applications. We’ll focus on understanding both technical and human factors behind successful DevOps transformations.

- Flexibility: The session is designed to adapt to different organizational structures and challenges.
- Conversations: Active participation and case-based discussions enhance collective learning.
- Literacy and Awareness: Gain a comprehensive overview of key DevOps principles and how they fit into your business.
- Focus on Human Behavior: DevOps success begins with people and processes before tools.
- Organizational Culture: Establishing a collaborative and growth-oriented culture is crucial for sustained success.
- Actionable Insights: Leave with practical next steps tailored to your organizational needs.

What not to expect from this workshop

- Prescriptions and formulas, rigid processes, step-by-step instructions
- Big overnight transformations
- Perfect solutions that work for everyone
- Extended technical discussions or deep focus on any specific engineering tool

8

We won't be prescribing rigid methodologies or offering cookie-cutter answers. Instead, we'll focus on principles that you can adapt to your business needs. Expect actionable advice, but remember—lasting change is gradual.

While we'll provide valuable insights, this workshop won't present a universal DevOps playbook. Instead, we focus on flexible, adaptive strategies.

- No Prescriptive Formulas: Every organization has different needs, and success depends on contextual adjustments.
- No Big Overnight Transformations: Effective DevOps adoption is incremental, focusing on continuous improvements.
- No Perfect Solutions: There's no magic bullet—instead, DevOps thrives on experimentation and refinement.
- No Extended Technical Deep Dives: This session is aimed at strategic decision-makers, keeping technical discussions at a high level.

DevOps for Executive Leadership (Week 7): Pipeline Metrics and Strategic Alignment

In this Week 7 session, we delve deeper into advanced DevOps practices that help large enterprises measure the impact of their pipeline on quarterly targets and strategic goals. Executives often want clear data that ties lead time, deployment frequency, or compliance gating success to high-level objectives. This deck provides an expanded look at pipeline metrics, including how to collect, interpret, and present them in dashboards that are accessible and valuable to leadership. We also review how Agile and Scrum ceremonies can incorporate these metrics so that the entire organization rallies around continuous delivery improvements.

Objectives for This Session

- Highlight **key pipeline metrics** that resonate with executive-level goals
- Demonstrate how to **instrument DevOps pipelines** for real-time data collection
- Explain ways to **map these metrics** to quarterly or strategic targets
- Show how **Agile and Scrum** incorporate metric-driven backlog tasks
- Provide a short **demo** of building out pipeline instrumentation and dashboards in preparation for Week 8's hands-on lab

1. **Key Pipeline Metrics**: We focus on advanced measurements (lead time, deployment frequency, compliance gating success, change failure rate, etc.) that matter at scale.

2. **Instrumenting DevOps Pipelines**: By adding logging or scanning steps, you generate the data executives want for real-time oversight.

3. **Mapping to Goals**: We detail how these metrics tie to broader organizational objectives, such as improved efficiency, reduced risk, or faster user-centric innovation.

4. **Agile + Metric-Driven Tasks**: Each pipeline expansion can be a backlog item, ensuring sprints deliver continuous pipeline enhancements.

5. **Demo**: Even though you don't have lab access this week, we illustrate how to

expand the pipeline to record and display these metrics. You'll do it hands-on in Week 8.

g on laptop with analytics overlay" to depict pipeline instrumentation.

Agenda

1. Recap of Key DevOps Concepts (High-Level)
2. Metrics That Matter
3. Mapping Metrics to Strategy
4. Agile Integration for Metric-Driven Tasks
5. Demo – Pipeline Instrumentation & Dashboards
6. Wrap-Up and Next Steps

11

Agenda Explanation

- We start with a short recap to anchor ourselves.
- **Section 1** explores advanced pipeline metrics in detail—beyond basic pass/fail.
- **Section 2** connects these metrics to organizational or quarterly objectives, bridging data with leadership's biggest concerns.
- **Section 3** shows how to embed metric improvements into agile ceremonies and backlogs.
- **Section 4** offers a pipeline instrumentation demo you can implement in your next lab (Week 8).
- Finally, we wrap up, clarifying next steps.

Recap of Key DevOps Concepts (High-Level)

- Shared ownership of pipeline success
- Continuous integration & short feedback loops
- Cross-functional squads bridging dev, ops, QA, security
- Agile sprints as a driver for small, iterative improvements

12

1. **Shared Ownership**: No single team or manager is solely responsible for pipeline stability; dev, ops, and security collaborate for consistent success.

2. **Continuous Integration & Feedback**: Early detection of issues leads to less rework. This principle underscores all advanced pipeline expansions.

3. **Cross-Functional Squads**: In large enterprises, bridging departmental silos is crucial. Everyone sees pipeline data, ensuring accountability.

4. **Agile Sprints**: Each sprint can incorporate devops tasks—particularly expansions that produce actionable metrics, helping the organization adapt quickly.

❖Metrics That Matter

- Lead time and deployment frequency
- Compliance gating success rate
- Operational efficiency indicators
- Tying these metrics to pipeline expansions

This section covers the **pipeline metrics** that large enterprises track to gauge devops maturity and success. We'll detail each metric's meaning, how it is captured, and why it's vital at an executive level.

Lead Time and Deployment Frequency

- Defining lead time (commit to production)
- Monitoring how often deploys happen
- Short lead time fosters nimble development
- Frequent deploys reduce big-batch risk

14

1. **Defining Lead Time**: A measure from code commit to live production. If it's short, your environment automation and gating are efficient.

2. **Monitoring Deploy Frequency**: Possibly daily or weekly releases. Smaller, more frequent merges lessen risk.

3. **Short Lead Time Fosters Nimble Development**: If an urgent feature or patch arises, you can deliver it quickly. That ties to user satisfaction or risk management.

4. **Frequent Deploys**: In a strict environment, smaller changes are safer. Large-batch updates carry bigger potential for compliance or security gaps.

Compliance Gating Success Rate

- Automated scans or policy checks in pipeline
- Counting how many runs pass/fail gating
- Demonstrates proactive risk management
- Real-time insight for leadership

15

1. **Automated Scans**: Security or compliance scanning ensures each commit meets internal or external regulations.

2. **Counting Pass/Fail**: Over time, a high pass rate means dev teams address security or compliance concerns proactively, while a spiky fail rate might highlight training or code-quality deficits.

3. **Proactive Risk Management**: Pipeline gating blocks risky changes early. This fosters trust among leadership that the org is not ignoring regulatory demands.

4. **Real-Time Insight**: Dashboards can show the pass/fail trend across squads, identifying which teams might need more help or updated scanning rules.

Operational Efficiency Indicators

- Time saved by automated tasks
- Self-service environment usage
- Resource utilization cost metrics
- Less wait time for dev & QA

16

1. **Time Saved by Automation**: Logging how many manual steps were replaced by pipeline tasks. This resonates with executives who want direct ROI.

2. **Self-Service Environment Usage**: If your pipeline spins up ephemeral test environments, track how often dev/QA request them and how quickly.

3. **Resource Utilization**: Show cost differences in ephemeral vs. always-on environments. This can tie to budget goals for lowering overhead.

4. **Less Wait Time**: By analyzing how quickly dev can get a stable test environment or how quickly pipeline gating completes, you measure the friction dev faces daily.

Tying Metrics to Pipeline Expansions

- Each metric improved by a specific pipeline step
- Example: gating pass rate → better scanning scripts
- Example: lead time → environment provisioning automation
- Agile backlog tasks drive each expansion

17

1. **Metric→Pipeline Step**: If you want gating pass rates to go up, you might refine scanning or add a new compliance check. If you want lead time to drop, environment tasks or automated merges are crucial.

2. **Example**: If gating pass rate is 70%, the backlog might have a user story for "Add thorough license scanning" or "Integrate new scanning library."

3. **Another Example**: If lead time is 8 days, user stories might include "Automate environment creation script to reduce manual ops wait."

4. **Agile Backlog**: Each pipeline improvement is a backlog item with acceptance criteria referencing the desired metric shift.

❖ Mapping Metrics to Strategy

- Setting quarterly targets
- Demonstrating ROI
- Aligning compliance gating with risk objectives
- Producing executive-friendly dashboards

This section illustrates how to **map devops metrics**—like lead time or gating pass rate—to the high-level strategic aims your organization follows each quarter or fiscal cycle. We'll address how to show the direct ROI from pipeline expansions, ensure compliance gating meets risk objectives, and generate dashboards for leadership.

Setting Quarterly Targets

- Identify high-priority metrics (speed, security, cost)
- Example: “Reduce lead time from 10 days to 5 by Q4”
- Track these in a devops strategy board
- Sprints break down each target into tasks

19

1. **Identify High-Priority Metrics**: Some orgs might pick lead time, others might pick gating compliance or cost-saving from ephemeral usage.

2. **Example**: If leadership sets a Q4 goal to reduce lead time by half, that becomes a guiding star for dev and ops expansions.

3. **DevOps Strategy Board**: A dedicated board that houses epics or high-level items for each metric target, subdivided into backlog tasks.

4. **Sprint-by-Sprint Approach**: Each iteration addresses one aspect, ensuring consistent progress rather than an all-at-once approach.

Demonstrating ROI

- Linking pipeline improvements to actual business wins
- Cost reduction from automated steps
- Lower incident counts or faster user adoption
- Present data in leadership reviews

20

1. **Linking Pipeline to Business Wins**: If your pipeline expansions allow a new product feature to ship 3 weeks earlier, highlight how that contributed to user adoption or revenue.

2. **Cost Reduction from Automation**: Show that removing manual tasks cut 20 developer hours per sprint, or that ephemeral environments saved thousands monthly in resource usage.

3. **Lower Incidents or Faster Adoption**: If your gating success soared, you might see fewer production rollbacks or better user sentiment.

4. **Present Data**: Summarize these findings in a concise executive deck or monthly check-in, showing pipeline expansions are not “technical overhead” but direct ROI.

Aligning Compliance Gating with Risk Objectives

- Executive concerns about regulatory fines or brand damage
- Setting thresholds for pipeline pass/fail
- Communicating gating success rates each month
- Linking gating data to risk dashboards

1. **Regulatory Fines or Brand Damage**: Large financial orgs worry about non-compliance leading to massive penalties. Gating pass/fail provides an early warning system.

2. **Setting Thresholds**: If a vulnerability has a CVSS score above X, the pipeline fails. This is a direct reflection of risk appetite.

3. **Communicating Gating Success**: If pass rates soared from 60% to 90%, you show that compliance is far more robust now. This fosters trust among leadership.

4. **Linking Gating Data to Risk Dashboards**: Risk committees might watch a real-time chart showing gating fails. Fewer fails or quicker fixes signals improved risk posture.

Producing Executive-Friendly Dashboards

- Summaries of lead time, gating pass, deployment frequency
- Minimal clutter, focusing on top 3–5 KPIs
- Regular automated updates
- Slack or Teams integration for any red flags

22

1. **Summaries of Key Metrics**: A single dashboard might show lead time trending, gating pass rates, and how many deployments occurred this week.

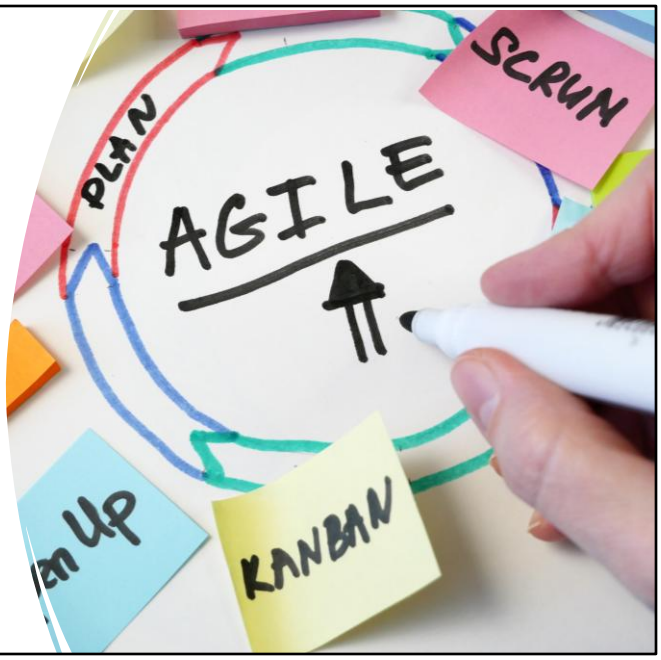
2. **Minimal Clutter**: Leadership wants quick insights. Overly technical logs can obscure the big picture. Provide clear, high-level charts or bullet metrics.

3. **Regular Automated Updates**: If your pipeline runs multiple times a day, the dashboard updates accordingly, ensuring near-real-time data.

4. **Alert Integration**: If gating pass rates suddenly dip or lead time spikes, auto-notifications can be sent to key stakeholders, letting them respond promptly.

❖ Agile Integration for Metric-Driven Tasks

- Using sprints to reduce lead time or improve gating
- Daily stand-ups referencing pipeline data
- Involving ops/security in backlog refinements
- Iterative approach to metric improvements



This section shows how agile ceremonies handle devops expansions that revolve around metrics. Each sprint can target a specific improvement—like shaving off lead time or boosting gating pass rate—and squads track progress daily.

lead time or gating expansions.”

Using Sprints to Reduce Lead Time

- Breaking down environment automation tasks
- Sprint planning with a lead time target
- Daily check for pipeline improvements
- Celebrating each small velocity gain

24

1. **Breaking Down Automation Tasks**: If environment provisioning is manual, tasks might be “Create Terraform scripts” or “Automate integration test environment.”

2. **Sprint Planning with Lead Time Target**: The backlog acceptance criterion might be “Lead time decreased by at least X hours.”

3. **Daily Check**: During stand-ups, squads confirm if new environment scripts or gating expansions are on track. Any pipeline block is escalated quickly.

4. **Celebrate Gains**: When lead time data shows improvement, highlight it in sprint reviews. This fosters excitement about DevOps expansions.

Daily Stand-Ups Referencing Pipeline Data

- If gating fails, dev fixes it within a day
- Security scanning triggers immediate tasks
- QA ensures test coverage expansions
- Everyone sees real-time logs in the backlog

25

1. **If Gating Fails**: The team addresses it that same day, not letting it linger. This approach builds habit around daily pipeline health.

2. **Security Scanning**: If scanning returns new vulnerabilities, the team might add a short-term fix story to the backlog.

3. **QA and Test Coverage**: QA can highlight flakey or incomplete tests that hamper gating success. Dev or QA quickly patch them.

4. **Real-Time Logs**: The pipeline's pass/fail logs tie into tasks in the backlog, so dev sees the environment variable or script that caused the gating to fail.

Involving Ops/Security in Backlog Refinements

- Ops proposes environment improvements
- Security sets gating thresholds as backlog stories
- Product owners weigh risk vs. user features
- Balanced sprint backlog ensures pipeline progress

26

1. **Ops Proposes Environment Tasks**: For instance, “Automate staging resource cleanup daily” might reduce cost, which is an efficiency metric.

2. **Security Sets Gating Thresholds**: “Implement fail if CVSS > 7” or “Add open source license scanning.” Each becomes a backlog item with acceptance criteria.

3. **Product Owners Weigh Risk vs. Features**: If gating tasks block user stories, the product owner can weigh the long-term risk savings or compliance demands.

4. **Balanced Sprint**: Some user-facing stories plus some devops expansions each sprint ensure no dimension of improvement is neglected.

Iterative Approach to Metric Improvements

- Each sprint commits to 1–2 devops expansions
- Retrospective measures new lead time or gating success
- Adjust backlog if results differ from expectations
- Over multiple sprints, pipeline matures significantly

27

1. **1–2 DevOps Expansions per Sprint**: This ensures constant pipeline improvement while user stories still proceed.

2. **Retrospective**: The team checks if the lead time or gating pass rate improved as planned, or if they overshot or undershot the target.

3. **Adjust Backlog**: If gating tasks took too long or an unexpected security tool problem arose, the team modifies or splits tasks for the next sprint.

4. **Pipeline Maturity Grows**: Over 3–4 sprints, incremental expansions yield a pipeline that is more robust, efficient, and secure.

❖ Demo – Pipeline Instrumentation & Dashboards

- Adding advanced metrics to an existing pipeline
- Instrumenting lead time & gating pass/fail
- Creating an executive-friendly dashboard
- Previewing what you'll implement in Week 8 lab

This demonstration shows how you can expand your pipeline to gather real-time data on lead time, gating pass/fail, or environment usage, then visualize it in a dashboard that leadership can easily interpret. We're not repeating the entire approach from earlier sessions, but focusing on advanced metrics.

Demo: Reviewing Pipeline with Basic Steps

STEP 1

- Open the minimal pipeline from prior labs/demos
- Notice build/test tasks but no advanced metrics
- Identify spots for logging lead time or gating results
- Confirm existing environment variables

1. **Open Minimal Pipeline**: Suppose the pipeline has only build and test stages. We see that we currently don't log the time difference or gating pass/fail in any structured format.

2. **Spots for Logging**: Right after "Checkout" or post-build, we can insert a script capturing commit timestamps. Similarly, after a compliance scanning step, we can parse pass/fail status.

3. **Existing Environment Variables**: For example, `Build.SourceVersion` or `System.PullRequest.SourceBranch`. We see how they might help us compute lead time or track gating references.

Demo: Instrumenting Lead Time

STEP 2

- Insert a script that calculates time from commit to pipeline start
- Storing the result in pipeline logs or a custom variable
- Optional step to post data to an analytics endpoint
- Validate the pipeline runs with new data displayed

30

1. **Insert Script**: In your YAML file, add a short block. For example, a Bash script or a PowerShell command that computes `(CurrentTime) – (CommitTime)`.

2. **Storing the Result**: We might `echo "LeadTimeMinutes=${calculatedValue}"` so the pipeline logs contain it, or we store it in a build variable for advanced dashboards.

3. **Optional Analytics Endpoint**: Some enterprises push data to an external system for long-term analysis or cross-team comparison.

4. **Validate**: We run the pipeline to confirm the logs show “LeadTimeMinutes= X.”

Demo: Capturing Gating Pass/Fail

STEP 3

- If scanning or compliance gating is used, parse the results
- Assign pass/fail to pipeline variables
- Summarize gating successes in the logs
- Potential Slack/Teams alert on failure

1. **Parsing Results**: The script for scanning typically returns an exit code or status. We interpret that, marking pass/fail.

2. **Assign to Pipeline Variables**: For instance, `echo "GatingPassed=true"` if it's good, or `GatingPassed=false` if not. Over time, dashboards can track how often gating fails.

3. **Log Summaries**: We might produce "Scan found 0 high-risk items. Gate passed." This is immediately visible to dev and ops.

4. **Slack/Teams Alert**: On a gating failure, a short message to a shared channel ensures immediate triage, reducing time wasted.

Demo: Executive-Friendly Dashboard

STEP 4

- Create or edit a new Azure DevOps dashboard
- Add widgets for lead time, gating success, deployment frequency
- Show how a manager could filter by branch or environment
- Present near-real-time data after each run

1. **Create/ Edit Dashboard**: Under “Dashboards,” add a panel specifically for leadership or product owners.

2. **Add Widgets**: If you have custom or marketplace widgets, you can graph lead time trending or gating pass rates over the last 10 runs.

3. **Filtering**: Some executives might only care about the main branch or production environment metrics. Enabling filters addresses that.

4. **Near-Real-Time**: Once the pipeline completes, the data updates. That fosters quick feedback loops for strategic discussions.

Demo: Observing the Pipeline Data

STEP 5

- Trigger a pipeline run to test new instrumentation
- Watch the logs produce lead time, gating pass/fail
- Refresh dashboard to see updated graphs
- Discuss linking these results to next sprint's backlog

1. **Trigger Run**: Doing a quick code commit or manual pipeline run.

2. **Watch Logs**: We see lines like "LeadTimeMinutes=14" or "GatingPassed=true." This confirms our instrumentation is working.

3. **Refresh Dashboard**: The lead time or gating success graph updates, letting anyone with access see real-time improvement or detect new issues.

4. **Discuss Linking to Backlog**: If we want to reduce lead time further, we might add tasks for environment scripts or scanning optimizations. Next sprint we tackle those.

Wrap-Up and Next Steps

- Summary of advanced pipeline metrics
- Demo recap: real-time lead time and gating logs
- Plan for Week 8 lab (hands-on expansions)
- Q&A or final discussions

34

1. **Summary**: We explored how to measure and present lead time, deployment frequency, compliance gating success, and operational efficiency data.

2. **Demo Recap**: We walked through adding instrumentation scripts, capturing pass/fail gating results, and setting up a leadership-friendly dashboard.

3. **Week 8 Lab**: You'll replicate these expansions and finalize an instrumented pipeline in a hands-on environment. This ensures your pipeline meets enterprise standards for data-driven devops.

4. **Q&A**: Answer any questions about connecting these metrics with agile backlog items, rolling up data for quarterly goals, or addressing specific compliance gating complexities.

Question 1

Which DevOps metric specifically tracks how quickly new code moves from commit to production?

A. Deployment frequency
B. Change failure rate
C. Mean time to restore
D. Lead time

This question checks whether you recall which metric focuses on speed from code commit to a live environment. Each possible answer is a known devops metric.

Question 1

Which DevOps metric specifically tracks how quickly new code moves from commit to production?

A. Deployment frequency
B. Change failure rate
C. Mean time to restore
D. **Lead time**

36

Explanation

Lead time measures the duration from the moment code is committed until it's running in production. This metric helps executives see how quickly the organization can respond to market needs or fix critical issues. Deployment frequency counts how many releases happen in a given period, change failure rate measures how often deployments cause incidents, and mean time to restore indicates how long it takes to recover from failures.

Question 2

How can compliance gating results best be integrated into an executive dashboard?

A. Only store them in raw logs for devops teams
B. Display a bar graph showing pass/fail trends over time
C. Hide compliance checks from leadership to avoid confusion
D. Summarize them annually in an audit report

37

Executives in large organizations often want real-time or near-real-time insight into how many pipeline runs pass compliance gating. This question covers the best approach to showcasing that data.

Question 2

How can compliance gating results best be integrated into an executive dashboard?

A. Only store them in raw logs for devops teams
B. **Display a bar graph showing pass/fail trends over time**
C. Hide compliance checks from leadership to avoid confusion
D. Summarize them annually in an audit report

38

Explanation
A bar graph or line chart reflecting pass/fail gating trends is the most transparent and helpful method for leadership. It allows them to see if compliance or security scanning is improving over sprints, linking devops expansions to risk management or regulatory goals.

Question 3

Which metric typically indicates the percentage of production deployments that lead to incidents or rollbacks?

A. Deployment frequency
B. Lead time
C. Change failure rate
D. Mean time to restore

This question focuses on the concept of how often releases cause issues. Each of the other metrics has a distinct definition, so identifying the correct one is key for measuring reliability.

Question 3

Which metric typically indicates the percentage of production deployments that lead to incidents or rollbacks?

A. Deployment frequency
B. Lead time
C. **Change failure rate**
D. Mean time to restore

40

Explanation

Change failure rate is the fraction of deployments that result in an incident or require a rollback. Deployment frequency is how often you release, lead time is speed from commit to production, and mean time to restore measures how quickly you recover once an incident happens.

Question 4

What best describes the benefit of connecting DevOps pipeline metrics to quarterly strategic goals?

A. It makes the pipeline data too technical for executives
B. It blocks user feature development
C. It helps leadership see direct ROI on pipeline improvements
D. It forces dev teams to slow down releases

This question covers the reason behind mapping devops metrics (like lead time or gating pass rates) to business objectives.

Question 4

What best describes the benefit of connecting DevOps pipeline metrics to quarterly strategic goals?

A. It makes the pipeline data too technical for executives
B. It blocks user feature development
C. **It helps leadership see direct ROI on pipeline improvements**
D. It forces dev teams to slow down releases

42

Explanation

When DevOps metrics (like reducing lead time or achieving higher compliance gating success) are tied to strategic goals (faster product rollouts, lower risk, cost savings), executives see tangible returns. This alignment fosters support and prioritization for devops tasks.

Question 5

Which approach helps large teams unify environment provisioning standards across multiple squads?

A. Every squad sets up their own ephemeral scripts
B. **Use a shared IaC repository and pipeline templates**
C. Rely on manual ops checklists in each product line
D. Only do environment provisioning once a quarter

This question addresses scaling devops environment practices in a large multi-squad environment.

Question 5

Which approach helps large teams unify environment provisioning standards across multiple squads?

A. Every squad sets up their own ephemeral scripts
B. **Use a shared IaC repository and pipeline templates**
C. Rely on manual ops checklists in each product line
D. Only do environment provisioning once a quarter

Explanation
Hosting a **shared Infrastructure as Code (IaC) repository** and pipeline templates ensures that each squad follows the same approach for environment creation. This reduces inconsistencies and fosters best practices across the organization. Relying on manual checklists or letting each squad do their own thing typically leads to fragmentation.