**1. Design and implement processes and communications**
  └ **1.2 Metrics and queries for DevOps**
    └ **1.2.1 Dashboards: cycle time, time to recovery, lead time**

1. What are cycle time, lead time, and time to recovery in DevOps, and how do they differ?
2. How do you configure dashboards in Azure DevOps to visualize these metrics?
3. Which built-in widgets are used to track cycle time, lead time, and time to recovery in AzDevOps?
4. What is the process for adding custom queries for metric visualization in Azure DevOps dashboards?
5. How can you monitor these metrics using GitHub Insights or GitHub Projects?
6. What actions can reduce cycle time and lead time in a DevOps workflow?
7. How do you configure alerts for abnormal values in these metrics?
8. How is time to recovery (MTTR) tracked in Azure DevOps and GitHub?
9. How should teams interpret trends and anomalies in these metrics for continuous improvement?
10. What are common pitfalls in configuring dashboards for DevOps metric

---

**1. What are cycle time, lead time, and time to recovery in DevOps, and how do they differ?**

- Cycle time: Time from starting work on a task to completion (e.g., first commit to deployment).
- Lead time: Time from when a request is made (work item created) to its delivery (deployment).
- Time to recovery (MTTR): Time taken to restore service after a production failure.
- Difference: Lead time includes all waiting; cycle time starts when actual work begins. MTTR focuses on incident recovery only.

---

**2. How do you configure dashboards in Azure DevOps to visualize these metrics?**

- Go to *Azure DevOps project > Dashboards > New Dashboard*.
- Add widgets such as Cycle Time, Lead Time, and Query Results.
- Configure each widget to target the correct team/project and backlog level.
- Save and share the dashboard as needed.

---

**3. Which built-in widgets are used to track cycle time, lead time, and time to recovery in Azure DevOps?**

- Cycle Time widget: Shows average time from "In Progress" to "Done."
- Lead Time widget: Displays time from creation to completion.
- Query Results widget: Can be customized to show MTTR based on resolved incidents or bug work items.

---

**4. What is the process for adding custom queries for metric visualization in Azure DevOps dashboards?**

- Go to *Boards > Queries > New Query*.
- Define criteria (e.g., State changes, tags for incidents).
- Save and run the query.
- Add the Query Results widget to a dashboard and link it to the saved query for visualization.

---

**5. How can you monitor these metrics using GitHub Insights or GitHub Projects?**

- *GitHub Insights* (in *GitHub Enterprise*) provides charts for lead time and cycle time per repo.
- *GitHub Projects* tracks issue and PR workflow states; custom fields and automation can approximate cycle/lead time.
- External tools (e.g., *Azure DevOps*, third-party analytics) can be integrated for MTTR tracking.

**6. What actions can reduce cycle time and lead time in a DevOps workflow?**

- Automate builds, tests, and deployments.
- Use small, incremental changes (shorter PRs).
- Improve code review and approval process speed.
- Minimize handoffs and blockers (clear ownership).
- Invest in high test coverage to catch issues early.

---

**7. How do you configure alerts for abnormal values in these metrics?**

- In *Azure DevOps*, use Analytics views or *Power BI* integration for threshold-based alerts.
- Configure dashboards with visual cues (color rules) for metric widgets.
- Use *Azure Monitor* or *Logic Apps* to trigger notifications for breached thresholds.

---

**8. How is time to recovery (MTTR) tracked in Azure DevOps and GitHub?**

- *Azure DevOps*: Use queries for work items marked as incidents/production bugs; calculate time from "Created" to "Resolved/Closed."
- *GitHub*: Track incidents with labeled issues or PRs; use timestamps to calculate recovery duration manually or via automation.

---

**9. How should teams interpret trends and anomalies in these metrics for continuous improvement?**

- Rising cycle/lead times indicate process bottlenecks or resource issues.
- Increasing MTTR signals gaps in incident response or recovery process.
- Regular review in retrospectives supports targeted process improvement.

---

**10. What are common pitfalls in configuring dashboards for DevOps metrics?**

- Using inconsistent work item states or definitions across teams.
- Not updating queries when workflows change.
- Focusing on vanity metrics without actionable insights.
- Failing to align dashboards to current team processes and priorities.