1. Design and implement processes and communications

└ 1.2 Metrics and queries for DevOps

└ 1.2.2 Metrics/queries for: planning, development, testing, security, delivery, operations

1. What key metrics should be tracked for planning in Azure DevOps?
2. What queries help monitor development progress and code health?
3. How do you set up testing metrics and visualize test pass rates?
4. What security metrics are essential in a DevOps pipeline?
5. How can delivery performance be measured using Azure DevOps queries?
6. What operational metrics should teams monitor post-deployment?
7. How do you build cross-area queries combining multiple DevOps stages?
8. What tools integrate with Azure DevOps to enhance metric reporting?
9. How can you automate metric collection and reporting across stages?
10. What are best practices for maintaining accurate, actionable queries?

---

1. What key metrics should be tracked for planning in Azure DevOps?

- Work item count (new, active, closed).
- Sprint/iteration velocity.
- Burndown/burnup charts.
- Forecast vs. actual completion.
- Backlog health (aging, blocked items).

---

2. What queries help monitor development progress and code health?

- Active pull requests and status.
- Code churn (lines added/removed).
- Commit frequency per developer/team.
- Open vs. resolved bugs.
- Code review completion rates.

---

3. How do you set up testing metrics and visualize test pass rates?

- Use *Test Plans > Runs > Query test results* (passed/failed/skipped).
- Add Test Results Trend widget to dashboards.
- Configure queries on test case outcomes over time.

---

4. What security metrics are essential in a DevOps pipeline?

- Number of open security bugs.
- Dependency vulnerability scan results.
- Secrets scan findings.
- Code scanning (e.g., static analysis) pass/fail counts.
- Compliance policy adherence.

---

5. How can delivery performance be measured using Azure DevOps queries?

- Deployment frequency (successful releases over time).
- Release success/failure rates.
- Average deployment duration.
- Rollback counts.
- Time from build to production deployment.

6. What operational metrics should teams monitor post-deployment?

- Incident counts and severity.
- Mean Time to Detect (MTTD).
- Mean Time to Recovery (MTTR).
- Service uptime and availability.
- Performance indicators (CPU, memory, latency).

---

7. How do you build cross-area queries combining multiple DevOps stages?

- Use Azure DevOps Analytics Views or custom queries linking work items, builds, and releases.
- Apply area paths, tags, or shared fields across queries to correlate stages.
- Combine results in *Power BI* or dashboards.

---

8. What tools integrate with Azure DevOps to enhance metric reporting?

- *Power BI* (advanced visualization).
- *Azure Monitor* and *Log Analytics* (telemetry integration).
- *Application Insights* (app-level metrics).
- Third-party tools like *SonarQube*, *WhiteSource* (code quality, security).

---

9. How can you automate metric collection and reporting across stages?

- Use *Azure Pipelines* tasks to export metrics post-stage.
- Set up scheduled *Analytics View* exports to *Power BI* or storage.
- Leverage REST APIs to pull and aggregate data.

---

10. What are best practices for maintaining accurate, actionable queries?

- Regularly review query definitions as processes evolve.
- Use consistent naming, tags, and fields.
- Validate data sources and update filters.
- Focus on actionable, team-relevant metrics.