3. Build and release pipelines

└ **3.1 Package management and testing strategy**

└ **3.1.4 Code coverage analysis**

---

1. What is code coverage and why is it important in DevOps pipelines?
2. How do you collect code coverage in Azure Pipelines?
3. What tools are commonly used for code coverage in .NET, Java, and JavaScript projects?
4. How do you publish and visualize code coverage reports in Azure DevOps?
5. How do you enforce code coverage thresholds in a pipeline?
6. How do you integrate code coverage analysis in GitHub Actions?
7. What are best practices for interpreting code coverage metrics?
8. How do you handle code coverage for integration and UI tests?
9. How do you troubleshoot missing or incomplete code coverage data?
10. How should teams use code coverage data to improve code quality?

---

1. What is code coverage and why is it important in DevOps pipelines?

Code coverage measures the percentage of source code executed by tests. It helps identify untested code, drives higher quality, and ensures changes are properly validated before deployment.

---

2. How do you collect code coverage in Azure Pipelines?

Add code coverage arguments to test commands (e.g., `/p:CollectCoverage=true` for .NET). Use tasks like `PublishCodeCoverageResults@1` to collect and upload coverage data for reporting.

---

3. What tools are commonly used for code coverage in .NET, Java, and JavaScript projects?

- .NET: *Coverlet, Visual Studio Code Coverage*
- Java: *JaCoCo, Cobertura*
- JavaScript: *Istanbul (nyc), Jest*

---

4. How do you publish and visualize code coverage reports in Azure DevOps?

Use the `PublishCodeCoverageResults` task with supported formats (*Cobertura, JaCoCo*). Results are visualized in the pipeline summary and the *Code Coverage* tab for detailed breakdowns.

---

5. How do you enforce code coverage thresholds in a pipeline?

Configure coverage tools to fail the build if coverage drops below a set threshold (e.g., using *Coverlet's* --threshold option). Add these checks in pipeline YAML to enforce quality gates.

---

6. How do you integrate code coverage analysis in GitHub Actions?

Add coverage tools to your workflow steps, generate a coverage report, and use actions (like `codecov/codecov-action` or `coverallsapp/github-action`) to upload results. Display coverage summary as a status check or comment.

---

7. What are best practices for interpreting code coverage metrics?

- Focus on critical code paths and business logic.
- High coverage does not guarantee quality; also review test depth and effectiveness.
- Investigate and reduce untested or unreachable code.

8. How do you handle code coverage for integration and UI tests?

- Instrument the application for integration/UI tests so coverage is measured across components.
- Merge results from multiple test types for comprehensive reporting.

---

9. How do you troubleshoot missing or incomplete code coverage data?

- Check test configuration and coverage tool integration.
- Ensure correct file paths and formats.
- Confirm that instrumented code is being executed and that reports are properly published.

---

11. **How should teams use code coverage data to improve code quality?**
    - Regularly review coverage trends.
    - Address gaps in high-risk or critical areas.
    - Use coverage as one input for quality, not as the only gate.
    - Prioritize meaningful tests over achieving arbitrary coverage targets.