3. Build and release pipelines
└─ **3.1 Package management and testing strategy**
└─ **3.1.3 Tests in pipelines: agents, result integration**

- What is the role of build/release agents in running pipeline tests?
- How do you select and configure agents for test execution?
- How are test tasks integrated in Azure Pipelines and GitHub Actions?
- How do you publish and visualize test results in Azure DevOps?
- How do you handle parallel test execution across agents?
- What formats are supported for test result files?
- How do you troubleshoot test execution issues on agents?
- How do you aggregate test results from multiple jobs or agents?
- How are flaky tests managed and reported in pipelines?
- What best practices ensure reliable test result integration in CI/CD?

---

1. What is the role of build/release agents in running pipeline tests?
Agents are the compute resources (VMs, containers, or hosted runners) that execute pipeline tasks, including building code, running tests, and publishing test results.

---

2. How do you select and configure agents for test execution?
- Choose agents based on OS, toolchain, and resource needs (e.g., MS-hosted vs. self-hosted).
- Configure agent pools in *Azure DevOps* or runners in *GitHub Actions*.
- Set demands and capabilities in pipeline YAML or settings.

---

3. How are test tasks integrated in Azure Pipelines and GitHub Actions?
Add built-in or custom test tasks (e.g., `VsTest`, `DotNetCoreCLI`, `PublishTestResults`) in *Azure Pipelines* YAML. In *GitHub Actions*, use setup and test actions, then upload results as artifacts.

---

4. How do you publish and visualize test results in Azure DevOps?
Use the `PublishTestResults@2` task to upload results in supported formats (*JUnit*, *TRX*). View test summaries, trends, and individual failure logs in the *Azure Pipelines* Test tab.

---

5. How do you handle parallel test execution across agents?
- Enable parallel jobs and test splitting in pipeline settings.
- Divide test suites by assemblies or test categories to distribute them across multiple agents, reducing total execution time.

---

6. What formats are supported for test result files?
Common supported formats include *JUnit*, *TRX*, *NUnit*, and *xUnit* XML. Use these formats to ensure results are properly parsed and displayed in *Azure DevOps* or *GitHub Actions*.

---

7. How do you troubleshoot test execution issues on agents?
- Review pipeline logs, agent diagnostic output, and test result files.
- Check for missing dependencies, incorrect paths, or environment mismatches.
- Use agent diagnostic commands as needed.

8. How do you aggregate test results from multiple jobs or agents?
Publish all test result files to the pipeline using the PublishTestResults task with the mergeTestResults option enabled. *Azure DevOps* will consolidate results for reporting.

---

9. How are flaky tests managed and reported in pipelines?
Flag flaky tests using test frameworks or pipeline annotations. *Azure DevOps* Test tab can highlight inconsistent tests. Regularly review and quarantine or fix unreliable tests.

---

10. What best practices ensure reliable test result integration in CI/CD?

- Standardize test result formats.
- Always publish results (even on failure).
- Isolate tests for reproducibility.
- Review integration logs after each run.
- Clean up environment between runs to avoid residue affecting results.