**Understand the delivery cadence and three types of triggers**

Release and stages make use of triggers. There are three types of triggers we recognize.

**Continuous deployment trigger**

You can set up this trigger on your release pipeline. Once you do that, your release pipeline will trigger every time a build completes and creates a new release.

**Scheduled triggers**

It allows you to set up a time-based manner to start a new release—for example, every night at 3:00 AM or 12:00 PM. You can have one or multiple daily schedules, but it will always run at this specific time.

**Manual trigger**

With a manual trigger, a person or system triggers the release based on a specific event. When it's a person, it probably uses some UI to start a new release. When it's an automated process, some events will likely occur. You can trigger the release from another system using the automation engine, which is usually part of the release management tool.

**release approvals**

In Azure DevOps, release approvals help enforce governance and control over deployments by requiring manual intervention before a release can proceed to the next stage. There are two types of approvals:

1. Pre-deployment Approvals – Require approval before a release is deployed to an environment.
2. Post-deployment Approvals – Require approval after deployment before proceeding to the next stage.

**release gates**

* **Incident and issues management**: Gate mechanisms ensure that deployment proceeds only if the required status for work items, incidents, and issues is met. For instance, deployment may be contingent on the absence of software bugs.
* **Approval integration with collaboration systems**: Integration with platforms like Microsoft Teams or Slack promotes communication with stakeholders for deployment approval, awaiting their response before proceeding.
* **Quality validation**: Gates can query metrics from tests on build artifacts, such as pass rate or code coverage, and deploy within specified thresholds to maintain quality standards.
* **Security scan on artifacts**: Gate mechanisms verify completion of security scans, such as anti-virus checks, code signing, and policy validation for build artifacts, ensuring compliance with security requirements before deployment.
* **User experience monitoring**: Leveraging product telemetry, gates validate that the user experience remains consistent with baseline standards, preventing deployment if regression is detected.
* **Change management integration**: Gates wait for change management procedures in systems like ServiceNow to conclude before proceeding with deployment.
* **Infrastructure health checks**: post-deployment, gates execute monitoring processes and validate infrastructure compliance against predefined rules, ensuring resource utilization and security standards are met.

**GitOps release strategy and recommendations**

