DevOps

Week 07

Murtaza Munawar Fazal

Continuous delivery (CD)

- Continuous delivery (CD) is a set of processes, tools, and techniques for rapid, reliable, and continuous software development and delivery.
- It means that continuous delivery goes beyond the release of software through a pipeline. The pipeline is a crucial component and the focus of this course, but continuous delivery is more.

Continuous delivery (CD)

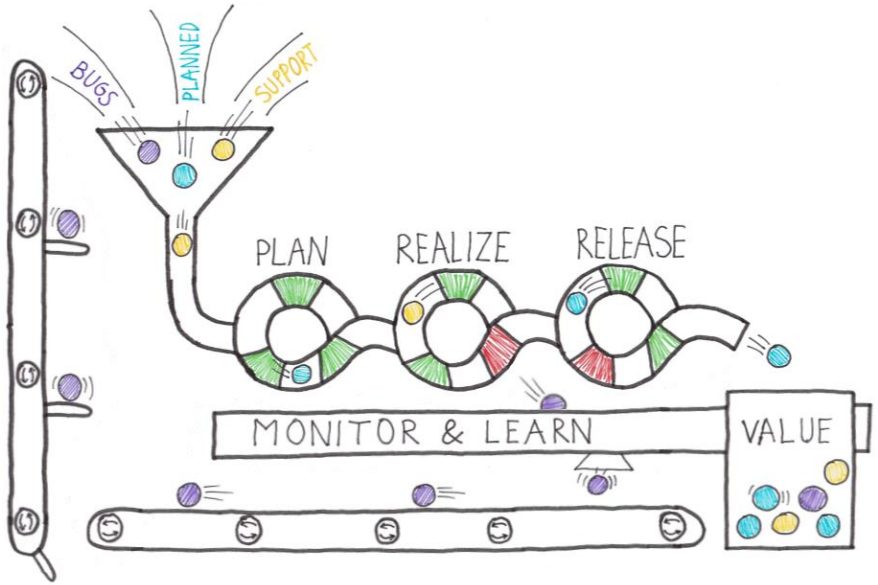
- Eight principles of continuous delivery:
  - The process for releasing/deploying software must be repeatable and reliable.
  - Automate everything!
  - If something is difficult or painful, do it more often.
  - Keep everything in source control.
  - Done means "released.“
  - Build quality in!
  - Everybody has responsibility for the release process.
  - Improve continuously.

Continuous delivery (CD)

- To deploy more often, we need to reconsider our:
  - Software architecture (monoliths are hard to deploy).
  - Testing strategy (manual tests don't scale well).
  - Organization (separated business and IT departments don't work smoothly), and so forth.

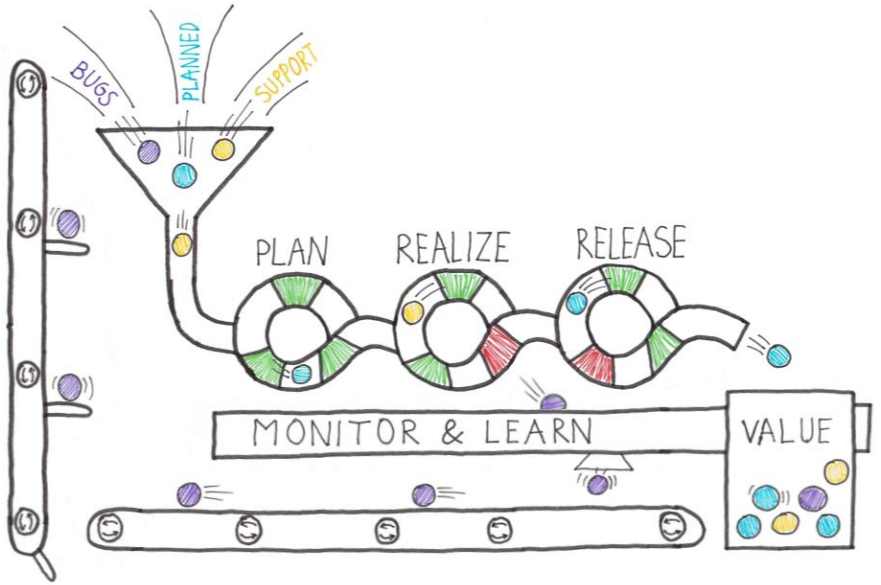
Why Continuous delivery (CD)

- Our customers demand working software, and they wanted it yesterday.
- If we can't deliver, they go to a competitor. And competition is fierce.
- We have competitors on our stack that deliver a best-of-breed tool for one aspect of the software we built.
- We need to deliver fast, and the product we make must be good. And we should do this with our software production being cheap and quality being high.
- To achieve this, we need something like Continuous Delivery.



Why Continuous delivery (CD)

- A piece of work is a marble. And only one part of the work can flow through the pipeline at once.
- Work must be prioritized in the right way.
- The pipeline has green and red outlets. These are the feedback loops or quality gates that we want to have in place.
- A feedback loop can be different things:
  - A unit test to validate the code.
  - An automated build to validate the sources.
  - An automated test on a Test environment.
  - Some monitor on a server.
  - Usage instrumentation in the code.
- If one of the feedback loops is red, the marble can't pass the outlet, and it will end up in the Monitor and Learn tray.
- The problem is analyzed and solved so that the next time a marble passes the outlet, it's green.



What is a Release?

- A release is a package or container containing a versioned set of artifacts specified in a release pipeline in your CI/CD process.
- It includes a snapshot of all the information required to carry out all the tasks and actions in the release pipeline, such as:
  - The stages or environments.
  - The tasks for each one.
  - The values of task parameters and variables.
  - The release policies such as triggers, approvers, and release queuing options.
- There can be multiple releases from one release pipeline (or release process).

Deployments

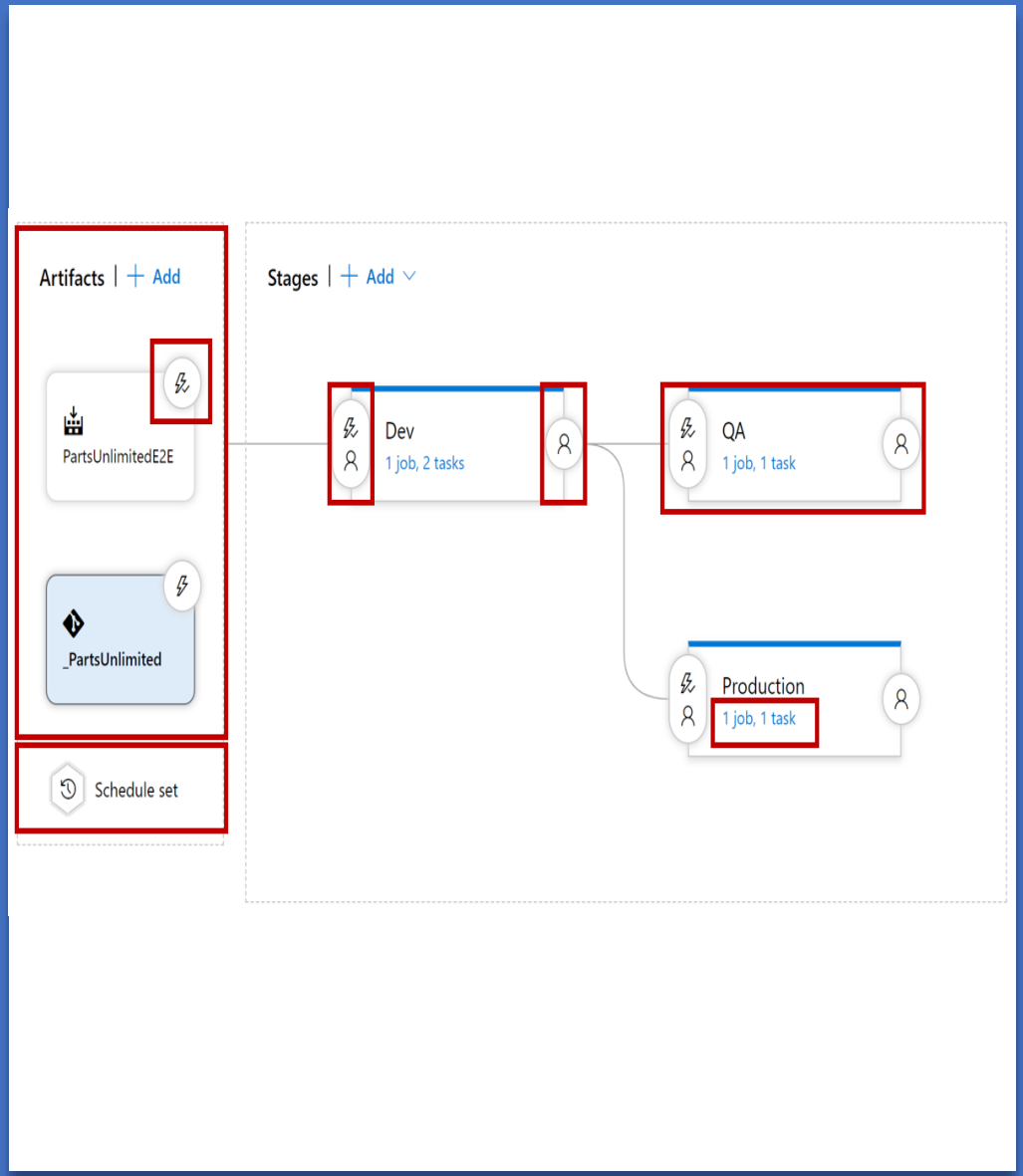
- Deployment is the action of running the tasks for one stage, which results in a tested and deployed application and other activities specified for that stage.
- Starting a release starts each deployment based on the settings and policies defined in the original release pipeline.
- There can be multiple deployments of each release, even for one stage.
- When a release deployment fails for a stage, you can redeploy the same release to that stage.

Release Process versus Release

- Release Process
  - The release pipeline (also known as release process) contains all the steps you walk through when you move your artifact from one of the artifact sources through the stages or environments.
  - The stage can be a development stage, a test stage, a production stage, or a stage where a specific user can access the application.
  - Part of your pipeline is the people who approve the release or the deployment to a specific stage. Also, triggers or schedules on which the releases execute, and the release gates, the automatic approvals of the process.
- Release
  - The release is an instance of the release pipeline. You can compare it with object instantiation.

Release Pipeline – Azure DevOps

- Components of Release Pipeline:
  - Artifacts
  - Trigger
  - Stages or Environments
  - Approval
  - Tasks



Release Pipeline – Azure DevOps

- Artifacts:
  - Artifacts can come from different sources.
    - The most common source is a package from a build pipeline.
    - Another commonly seen artifact source is, for example, source control.
- Trigger: the mechanism that starts a new release.
  - A trigger can be:
    - A manual trigger, where people start to release by hand.
    - A scheduled trigger, where a release is triggered based on a specific time.
    - A continuous deployment trigger, where another event triggers a release. For example, a completed build.

Release Pipeline – Azure DevOps

- Another component of a release pipeline is stages or sometimes called environments.
- It's where the artifact will be eventually installed.
- For example, the artifact contains the compiled website installed on the webserver or somewhere in the cloud. You can have many stages (environments); part of the release strategy is finding the appropriate combination of stages.

Release Pipeline – Azure DevOps

- Another component of a release pipeline is approval.
- People often want to sign a release before installing it in the environment.
- In more mature organizations, this manual approval process can be replaced by an automatic process that checks the quality before the components move on to the next stage.

Release Pipeline – Azure DevOps

- The tasks are the steps that need to be executed to install, configure, and validate the installed artifact.

Artifact

- An artifact is a deployable component of your application. These components can then be deployed to one or more environments.
- In general, the idea about build and release pipelines and Continuous Delivery is to build once and deploy many times.
- It means that an artifact will be deployed to multiple environments. The artifact should be a stable package if you want to achieve it.
- The configuration is the only thing you want to change when deploying an artifact to a new environment.
- The contents of the package should never change. It's what we call immutability. We should be 100% sure that the package that we build, the artifact, remains unchanged.

Build / Release Pipelines

- The build pipeline compiles, tests, and eventually produces an immutable package stored in a secure place (storage account, database, and so on).
- The release pipeline then uses a secure connection to this secured place to get the build artifact and do extra actions to deploy it to an environment.
- The significant advantage of using a build artifact is that the build produces a versioned artifact.
- The artifact is linked to the build and gives us automatic traceability.
- We can directly link our version control to our release pipeline.
- The release is related to a specific commit in our version control system. With that, we can also see which version of a file or script is eventually installed. In this case, the version doesn't come from the build but from version control.
- Consideration for choosing a version control artifact instead of a build artifact can be that you only want to deploy one specific file. If you don't need to run more actions before using this file in your release pipeline, creating a versioned package (build artifact) containing only one file doesn't make sense.