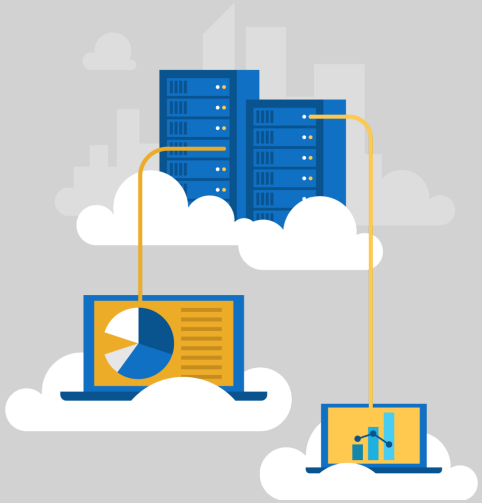
Microsoft

AZ-400.3 Module 1: Design a Release Strategy

Learning Objectives

- Differentiate between a release and a deployment
- Define the components of a release pipeline
- Explain things to consider when designing your release strategy
- Classify a release versus a release process, and outline how to control the quality of both
- Describe the principle of release gates and how to deal with release notes and documentation
- Explain deployment patterns, both in the traditional sense and in the modern sense
- Choose a release management tool

Lesson 01: Introduction to Continuous Delivery



What is Continuous Delivery?

Continuous Delivery is a software development discipline where you build software in such a way that the software can be released to production at any time

“Martin Fowler”

Traditional IT Company

- A fair amount of hotfixes and change requests for production
- Many scope changes during a project
- Lots of unplanned work due to technical debt (environment drift, poor quality, handoffs)
- Business involved but not attached to IT

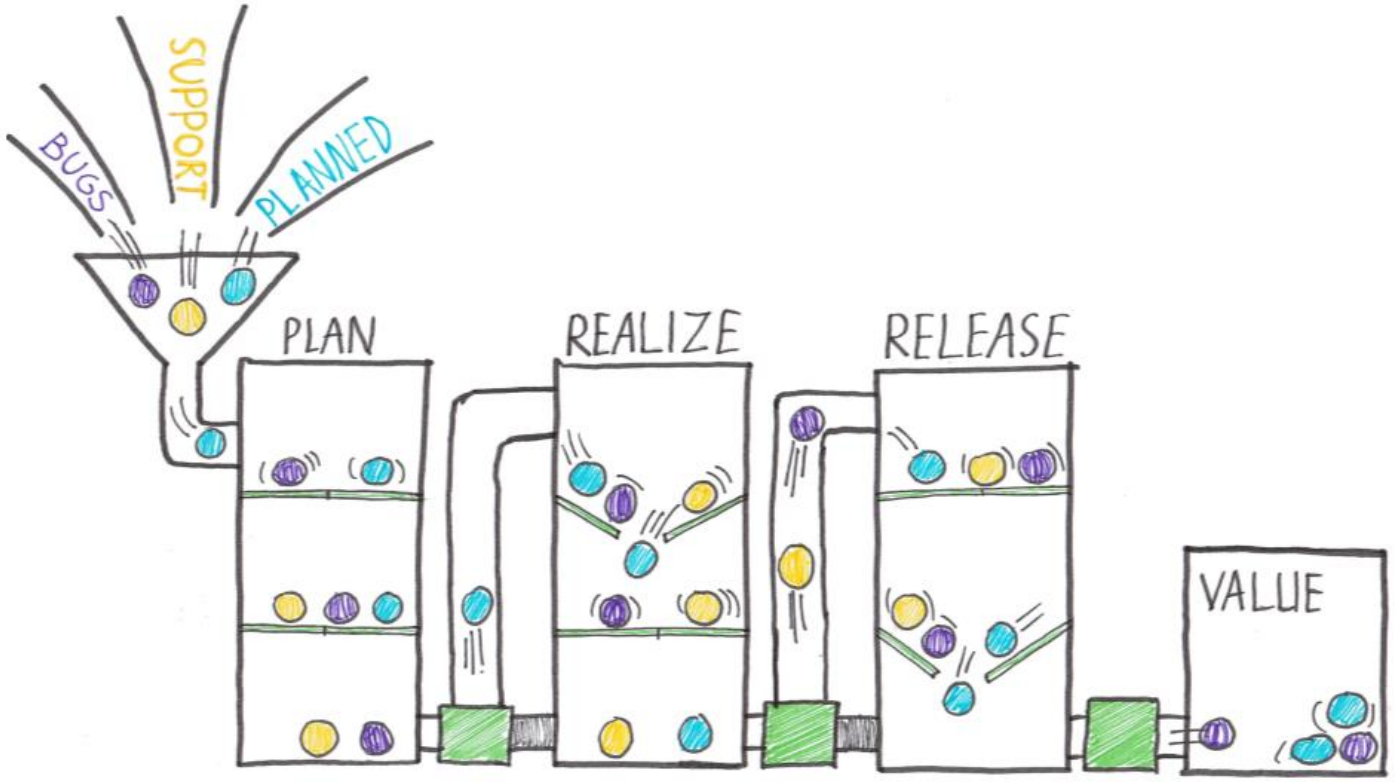
Why is this the case?

- IT should support core business and should be cheap
- Change is dangerous – should be kept under strict control
- Software is of poor quality and should be thoroughly tested
- Business works with customers and fixed functionality and time
- Computer power expensive and must be used optimally
- Customers don’t want to wait for the next release so use hotfixes as a shortcut or change their requirements

The result

- Development motivated by change from business pressure
- IT motivated to change nothing to prevent introducing risk
- Result is long cycles and large batch sizes that pile on risk

Traditional IT Companies



Times have changed

Companies need to become

Better

Quality needs to be high. Applications need to be secure. Code needs to be maintainable

Faster

Customers demand features. They want it fast, otherwise they go to the competitor

Cheaper

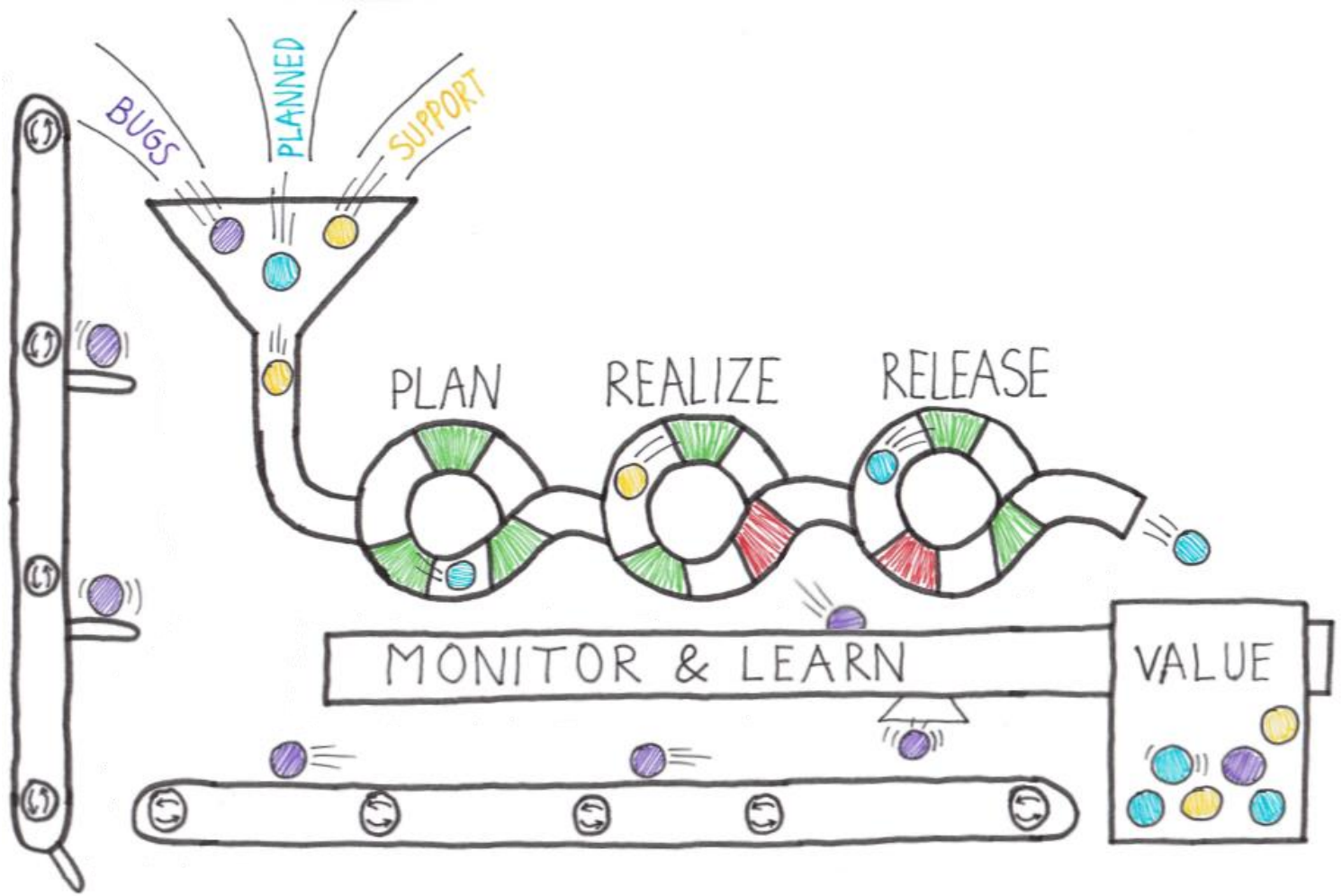
Competition is fierce and we are competing with the neighbor next door

IT is the business

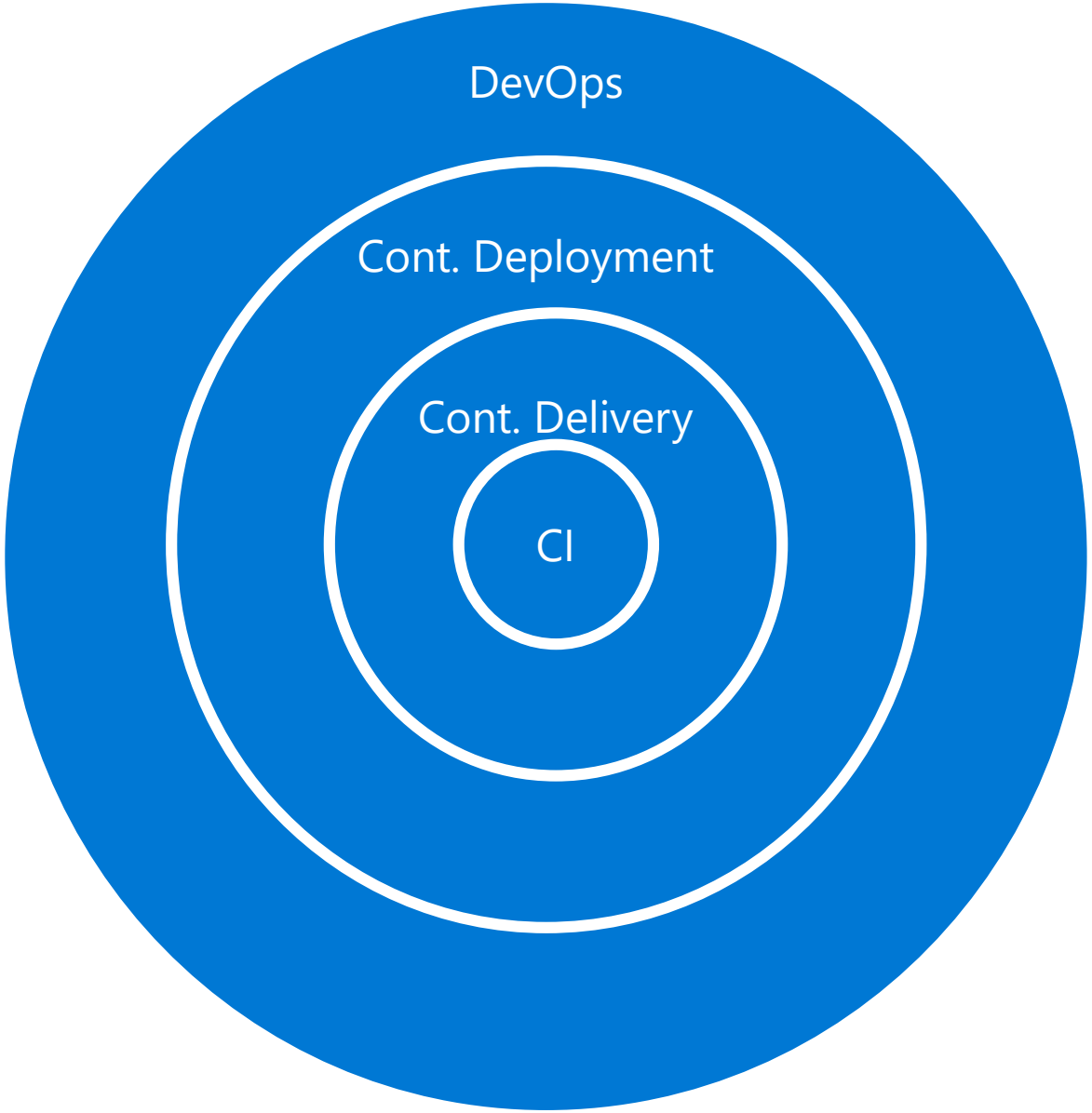
Continuous Delivery

- Continuous Delivery is delivering on the Agile Promise
- Bottlenecks shifted after implementing Agile
- Problem is within Delivery
  - Separation of Dev and Ops
  - Dev looks for change
  - Ops looks for Stability
  - This blocks delivery
- By implementing DevOps and Continuous Delivery we should be able to push out changes on demand !

Continuously delivering value



Continuous Integration, Delivery, Deployment and DevOps



The Eight Principles of Continuous Delivery

The process for releasing/deploying software MUST be repeatable and reliable.

Automate everything!

If somethings difficult or painful, do it more often.

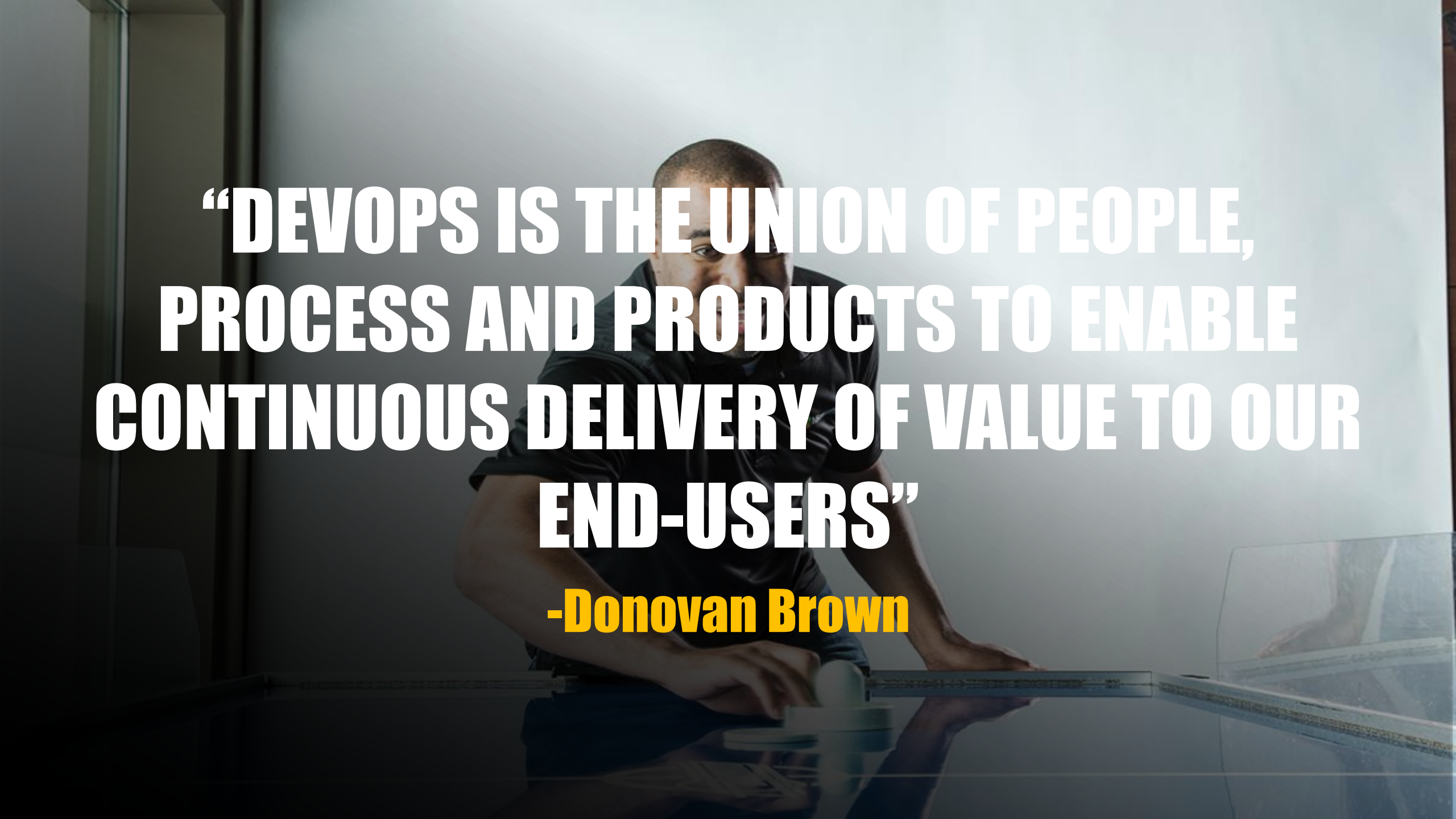
Keep everything in source control.

Done means “released”.

Build quality in!

Everybody has responsibility for the release process.

Improve continuously.


“DEVOPS IS THE UNION OF PEOPLE, PROCESS AND PRODUCTS TO ENABLE CONTINUOUS DELIVERY OF VALUE TO OUR END-USERS”
-Donovan Brown

Discussion

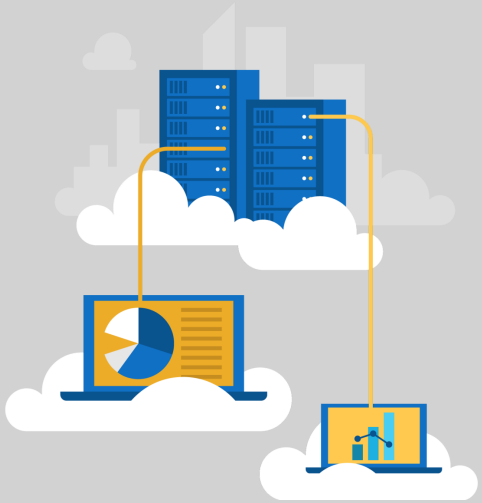
What are your bottlenecks?

Have a discussion about the need for Continuous Delivery in your organization and what blocks the implementation

Topics you might want to discuss are:

- Does your organization need Continuous Delivery?
- Do you use Agile/Scrum?
- Is everybody involved or only the Dev departments?
- Can you deploy your application multiple times per day? Why or why not?
- What is the main bottleneck for Continuous Delivery in your organization?
- The Organization
- Application Architecture
- Skills
- Tooling
- Tests
- other things?

Lesson 02: Release Strategy Recommendations



Release And Deployments

- Release and deployment are often coupled
- Release is not the same as a deployment
- Separate that in functional and technical release
  - Functional release is exposing features to customers
  - Technical release is deploying functionality

Feature toggles

What is it?

- Mechanism to separate feature deployment from feature exposure
- A.k.a. feature flippers, feature flags, feature switch, conditional feature, etc.

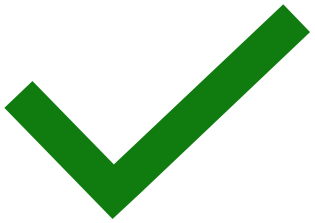
Why do you need it?

- It enables you to give control back to the business on when to release the feature
- Enables A/B testing, canary releases and dark launching
- It provides an alternative to keeping multiple branches in version control
- Enables change without redeployment

In tooling it is similar but different

Release

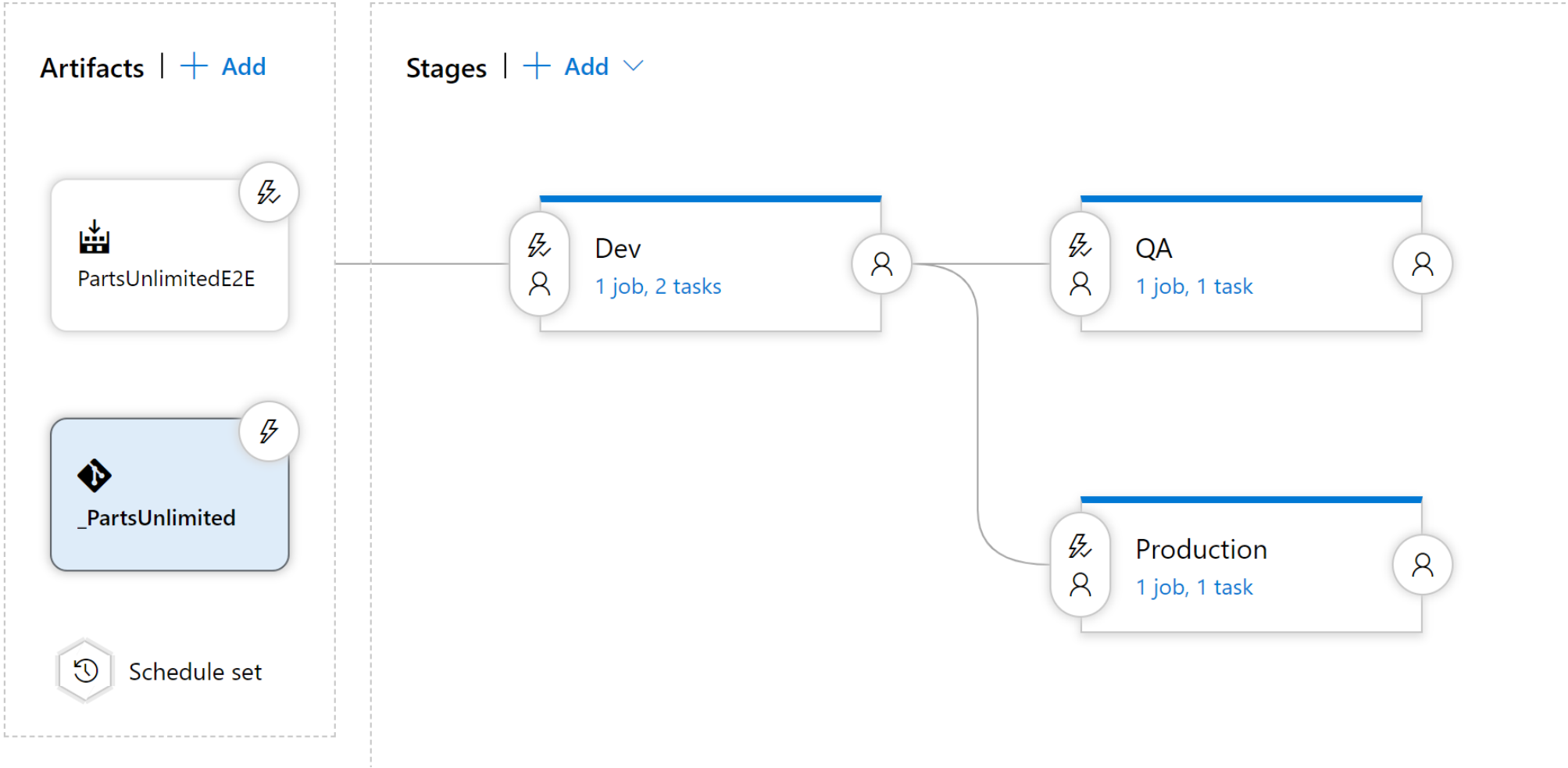
Container with versioned artifacts, pipeline, approvals, stages, variables



Deployment

All the actions and tasks that need to be executed

Components in a release pipeline



Artifact Sources

- Build Artifacts
- Package repositories
- Container Repositories
- Files
- Source Control

Considerations for choosing the rights artifact source

TRACEABILITY
AND AUDITABILITY

IMMUTABILITY

VERSIONING

READING
MATERIAL

Selecting an artifact source

Deployment Stages

What is a stage?

- Many different terms used in various release tools
- Stages/environment
- It is a dedicated place where people can fulfill their needs

What is your target environment?

Long-lived or short-lived environments

Purpose of an environment

Feature release and bugfix

Discussion

What deployment stages would you define for your organization?

- Have a discussion about what deployment stages you recognize in your organization?
- Consider the following things:
- Is your stage long lived or short lived?
- What is the purpose of this specific stage?
- Who is going to use it?
- Is your target application overwriting an existing one would always be a fresh install?
- Do you need a new stage for bug fixes?
- Do you need an isolated environment with encrypted data or disconnected from a network?
- Can you afford downtime?
- Who is the owner of the stage? Who can apply changes?

Setting up deployment stages

Delivery cadence - three types of triggers

CONTINUOUS DEPLOYMENT TRIGGER

SCHEDULED TRIGGER

MANUAL TRIGGER

Considerations for cadence

- Do we want/need to deploy every day?
- What is your target environment?
- Is it used by one team or is it used by multiple teams?
- Who are the users? Do they want a new version multiple times a day?
- How long does it take to deploy?
- Is there downtime? What happens to performance? Are users impacted?

Selecting your Delivery and Deployment cadence

Release Approvals

Release approvals are not to control *how*, but control *if* you want to deliver multiple times a day.

Manual Approvals help in building trust about the automated release process

Considerations for manual approvals

- What do we want to achieve with the approval?
- Who needs to approve?
- When do you want to approve?

Release Gates

Release gates give you additional control over the start and completion of the deployment pipeline. They can usually be set up as a pre-deployment and post-deployment condition and can do validation with other automated systems until specific requirements are verified.

Release Gate Examples

INCIDENT AND ISSUES MANAGEMENT

NOTIFICATION OF USERS BY INTEGRATION WITH COLLABORATION SYSTEMS

QUALITY VALIDATION

SECURITY SCAN ON ARTIFACTS

USER EXPERIENCE RELATIVE TO BASELINE

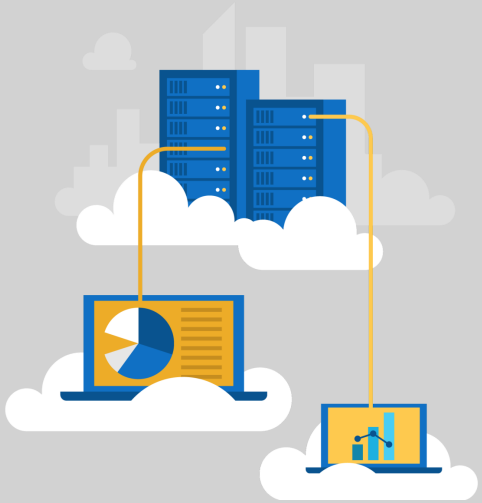
CHANGE MANAGEMENT

INFRASTRUCTURE HEALTH

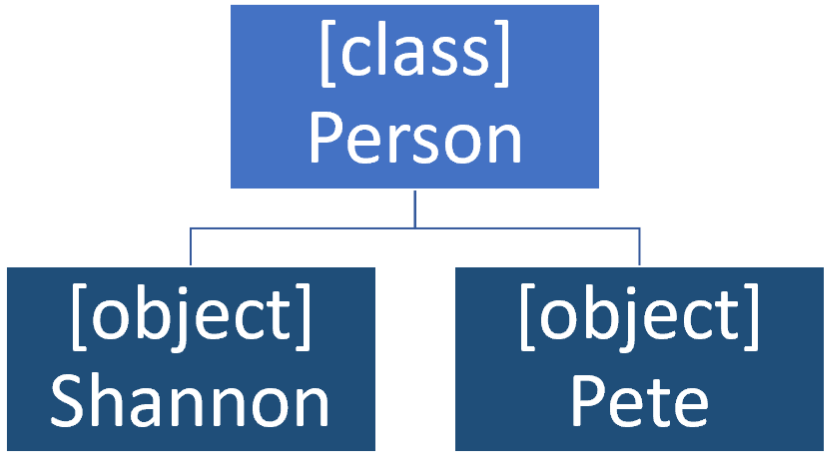
Setting up Manual Approvals

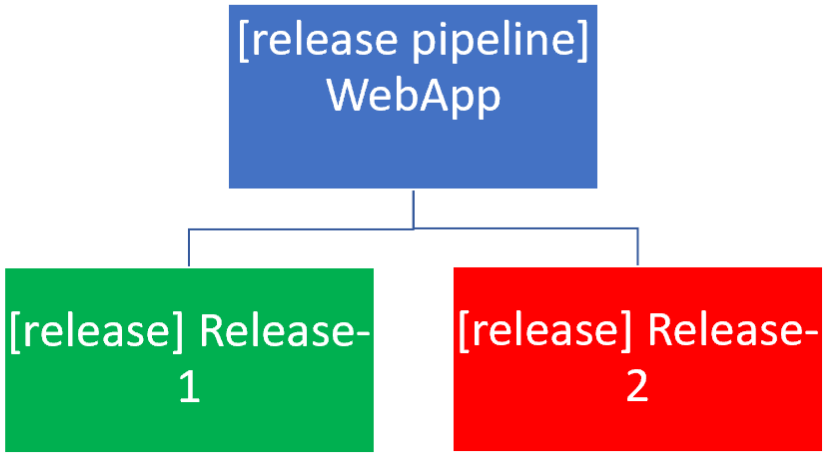
Setting up a Release Gate

Lesson 03: Building a High Quality Release Pipeline



Release and Release Process





Release and Release Process

The release pipeline contains all the steps that you walk through when you move your artifact, that comes from one of the artifact sources

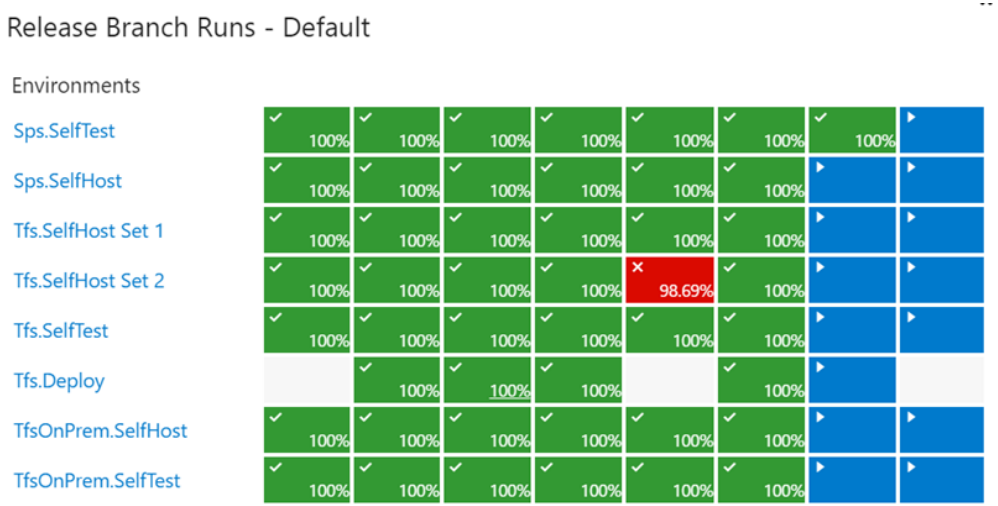
A release is a package or container that holds a versioned set of artifacts specified in a release pipeline in your CI/CD process. It includes a snapshot of all the information required to carry out all the tasks and actions in the release pipeline, such as the stages (or environments), the tasks for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

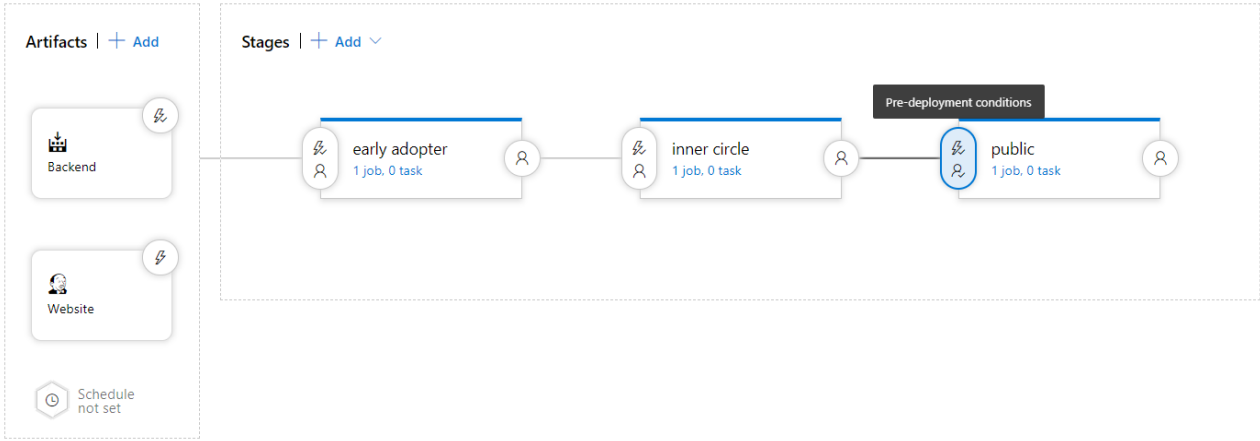
How to measure quality of your release process

Visualize your release process

Symptoms of broken process (every second day, only after rerun, never ending up in last stage)

Dashboard widgets





Quality Gates

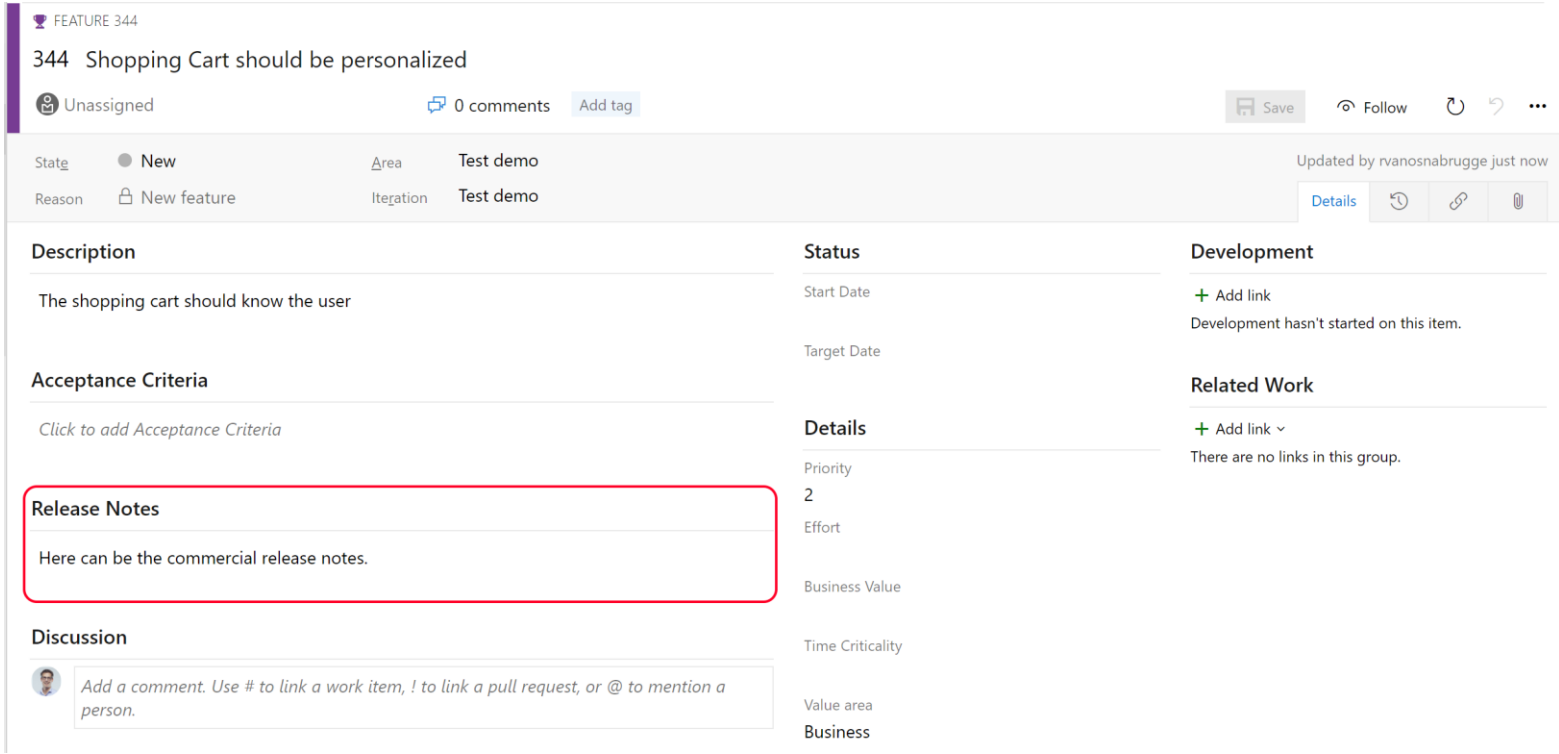
A quality gate is the best way to enforce a quality policy in your organization. It's there to answer one question: can I deliver my application to production or not?

Using Release Gates to protect quality

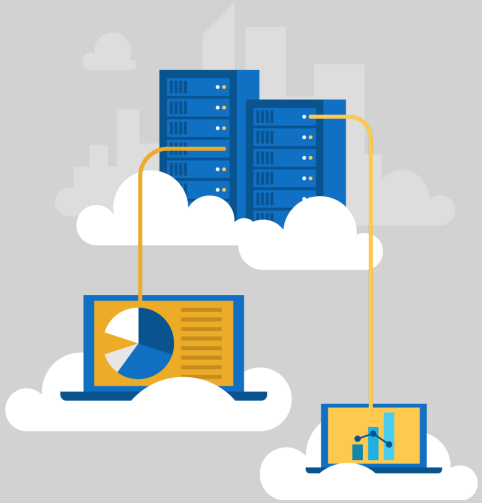
- No new blocker issues
- Code coverage on new code greater than 80%
- No license violations
- No vulnerabilities in dependencies
- No new technical debt introduced
- Compliance checks
- Are there work items linked to the release?
- Is the release started by someone else as the code committer?
- Is the performance not affected after a new release?

Release notes and Documentation

- Technical or Functional Documentation?
- Where to store Documentation
  - Document Store
  - Wiki
  - In the code base
  - In a Work Item



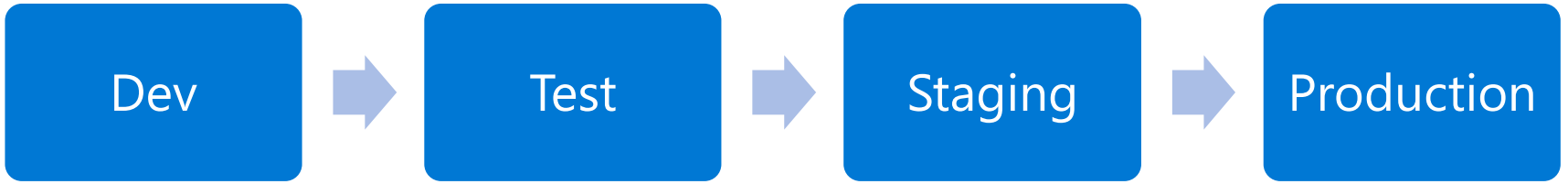
Lesson 04: Choosing a Deployment Pattern



Deployment patterns

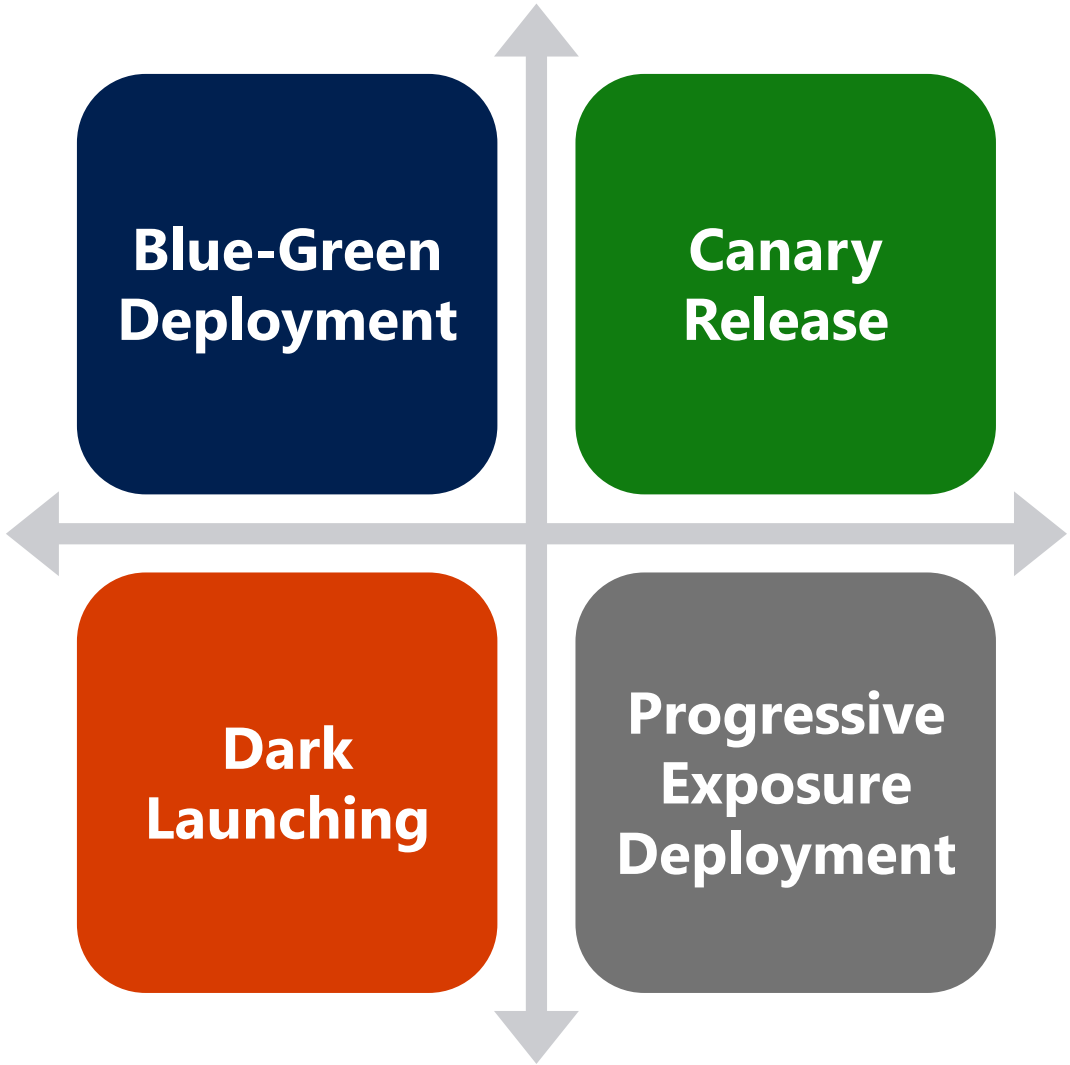
A deployment pattern is a way of how you choose to move your application to production.

Traditional Deployment Patterns (DTAP)

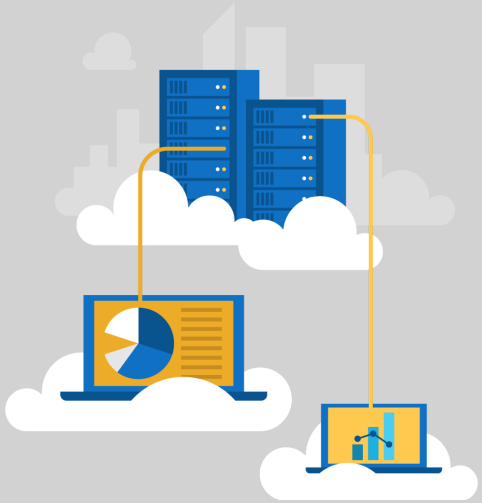


Modern Deployment Patterns

Modern Deployment Patterns



Lesson 05: Choosing the Right Release Management Tool



Release Management Tools

Tools that can do Build and Continuous Integration and Deployment

Tools that can do Release Management

Considerations for choosing

- Artifacts and Artifact source
- Triggers and Schedules
- Approvals and gates
- Stages
- Build and Release Tasks
- Traceability, Auditability and Security

Release Management Tools

Jenkins
Circle CI
Azure DevOps Pipelines
GitLab Pipelines
Atlassian Bamboo
XL Deploy/XL Release

GitLab

Azure Pipelines

Jenkins

circleci

Building a release strategy

In this exercise, you need to design a release strategy for a PartsUnlimited. Based on the concepts you learned, your assignment is to draw a picture or design a release pipeline in Azure DevOps where you think about the following things.

Choosing the rights artifact source
Which deployment stages do you choose
Release triggers and deployment triggers
Manual Approvals and/or Release Gates
Applicable deployment pattern(s)

Wrap up

- Differentiate between a release and a deployment
- Define the components of a release pipeline
- Explain things to consider when designing your release strategy
- Classify a release versus a release process, and outline how to control the quality of both
- Describe the principle of release gates and how to deal with release notes and documentation
- Explain deployment patterns, both in the traditional sense and in the modern sense
- Choose a release management tool

Module 1: Review Questions

1. Would adding a feature flag increase or decrease the cyclomatic complexity of the code?
2. You plan to slowly increase the traffic to a newer version of your site. What type of deployment pattern is this?
3. What can you use to prevent a deployment in Azure DevOps when a security testing tool finds a compliance problem?