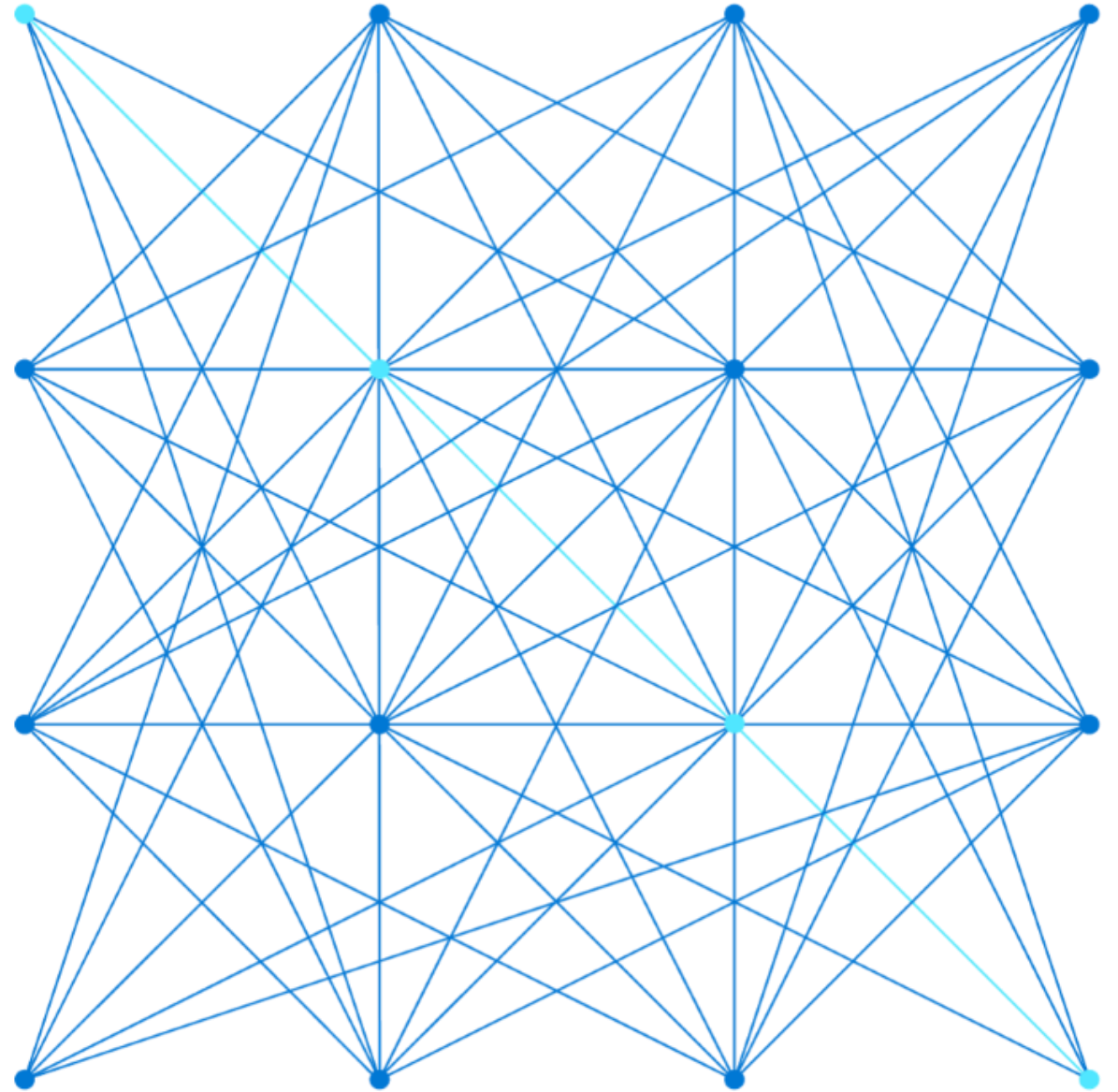
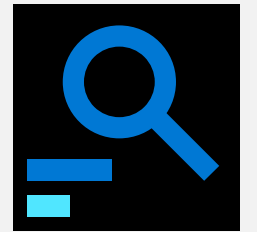


AZ-400.00

Module 10: Designing a Release Strategy



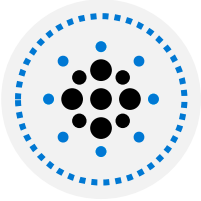
Lesson 01: Module overview



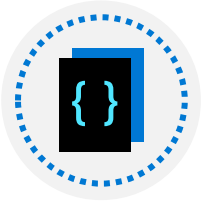
Module overview



Lesson 1: Module overview



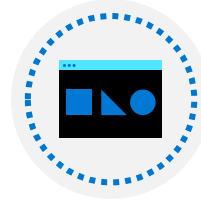
Lesson 2: Introduction to continuous delivery



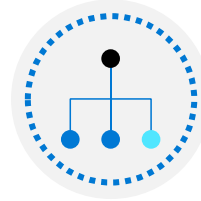
Lesson 3: Release strategy recommendations



Lesson 4: Building a high-quality release pipeline



Lesson 5: Choosing the right release management tool



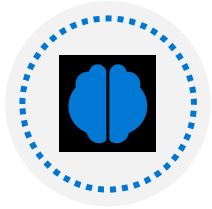
Lesson 6: Labs



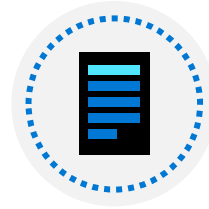
Lesson 7: Module review and takeaways

Learning objectives

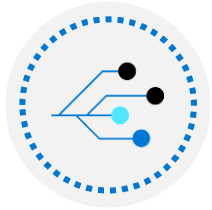
After completing this module, students will be able to:



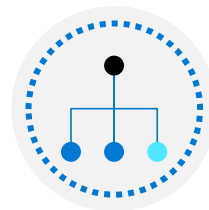
Differentiate between a release and a deployment



Describe the principle of release gates and how to deal with release notes and documentation



Define the components of a release pipeline



Choose a release management tool

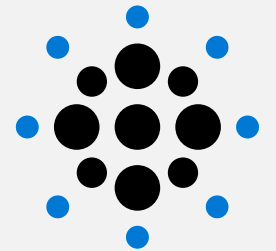


Explain things to consider when designing your release strategy



Classify a release versus a release process, and outline how to control the quality of both

Lesson 02: Introduction to continuous delivery



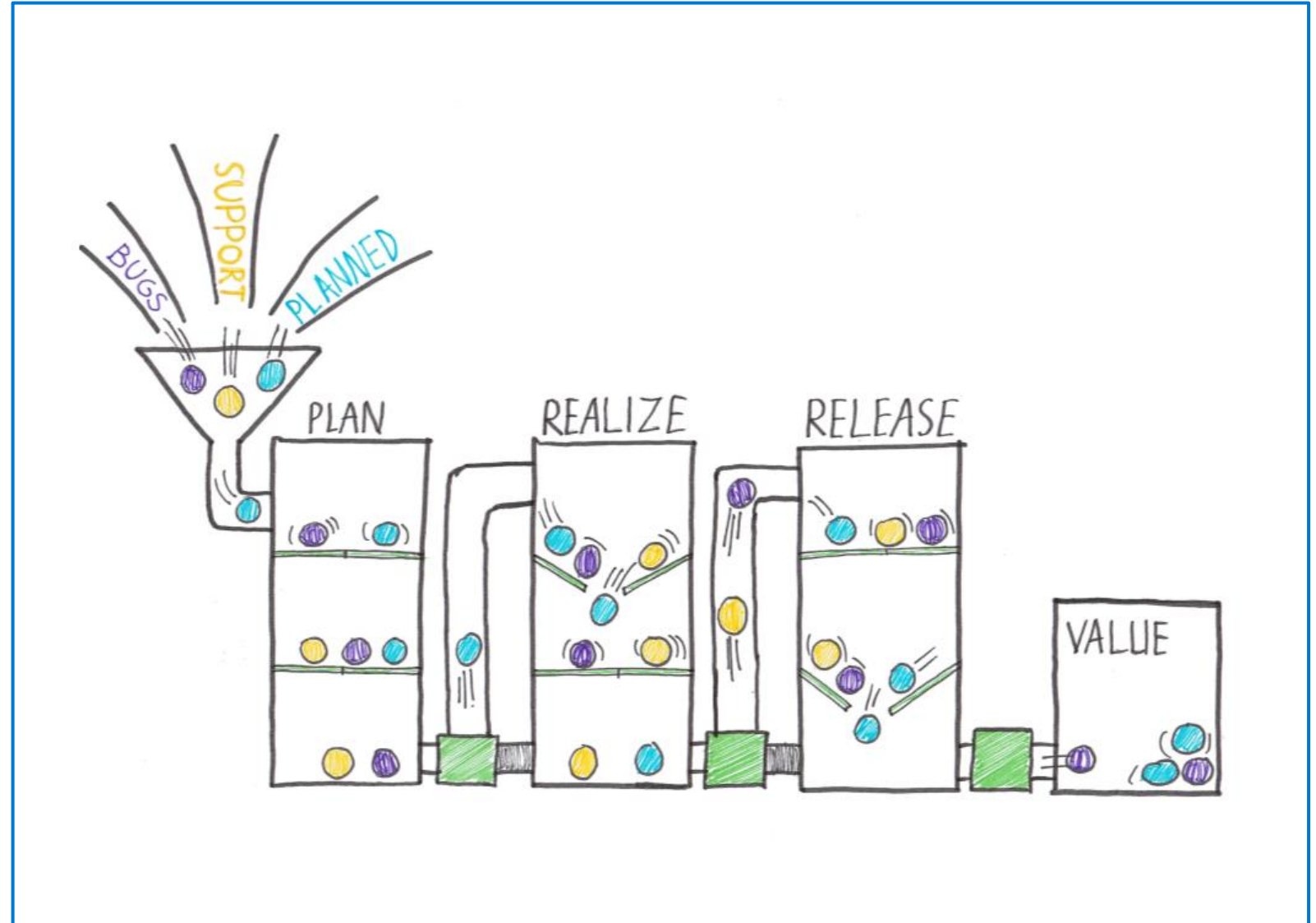
Traditional IT development cycle

A fair amount of hotfixes and change requests for production

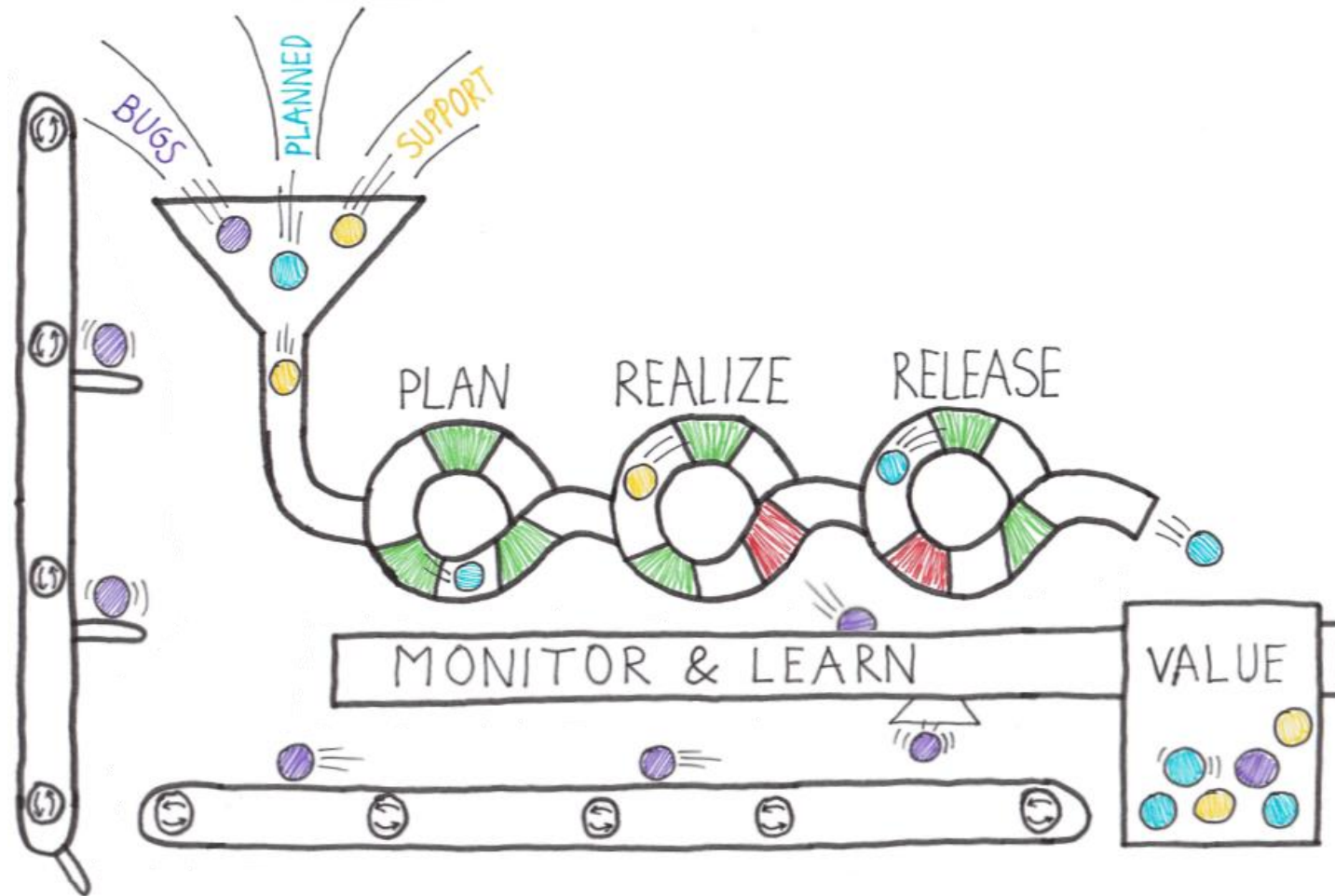
Many scope changes during a project

Lots of **unplanned work** due to technical debt (environment drift, poor quality, handoffs)

Business involved but not attached to IT



Moving to continuous delivery

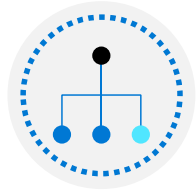


What is continuous delivery?

The eight principles of continuous delivery:



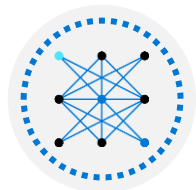
The process for releasing/deploying software
MUST be repeatable and reliable



Keep everything in source control



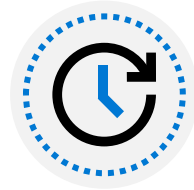
Everybody has responsibility for the
release process



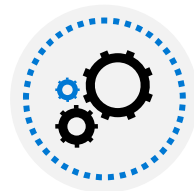
Automate everything!



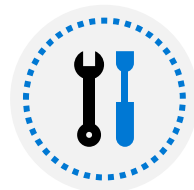
Done means “released”



Improve continuously

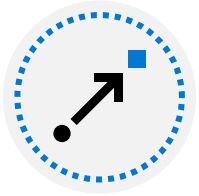


If something is difficult or painful, do it
more often

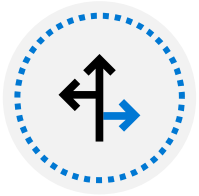


Build quality in!

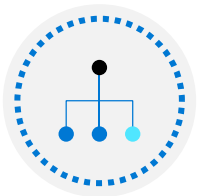
Releases and deployments



Release and deployment are often coupled



Release is not the same as a deployment



Separate functional release from technical release:

Functional release is exposing features to customers

Technical release is deploying functionality

Discussion – the need for continuous delivery in your organization

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

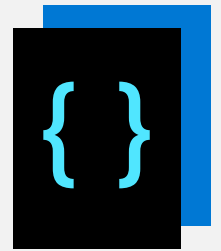
Skills

Tooling

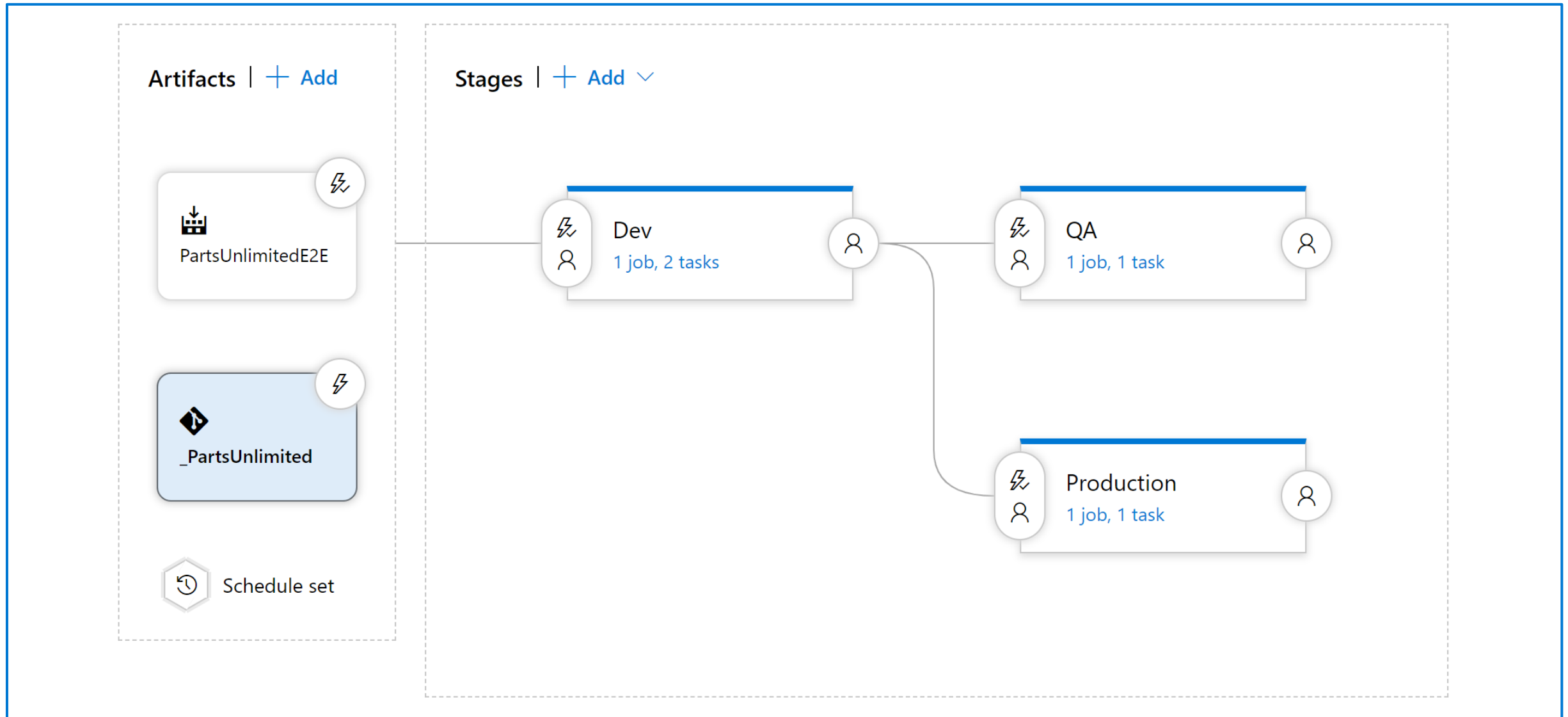
Tests

Other things?

Lesson 03: Release strategy recommendations



Release pipelines



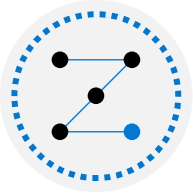
az-p-releasePipeline?

D-mw-Release Pipeline demo

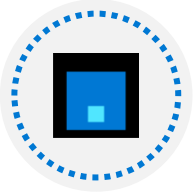
Artifact sources (#mw – In this context, artifacts are not just “Azure Artifacts”)



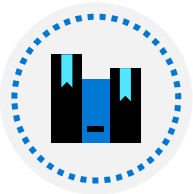
Build artifacts



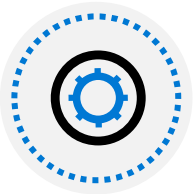
Package repositories



Container repositories



Files



Source control

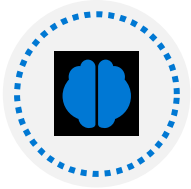
Considerations for choosing the appropriate artifact source



Traceability and auditability



Immutability



Versioning

Demonstration: selecting an artifact source

DEMO

Considerations for deployment to **stages**

Do we want/need to deploy every day?

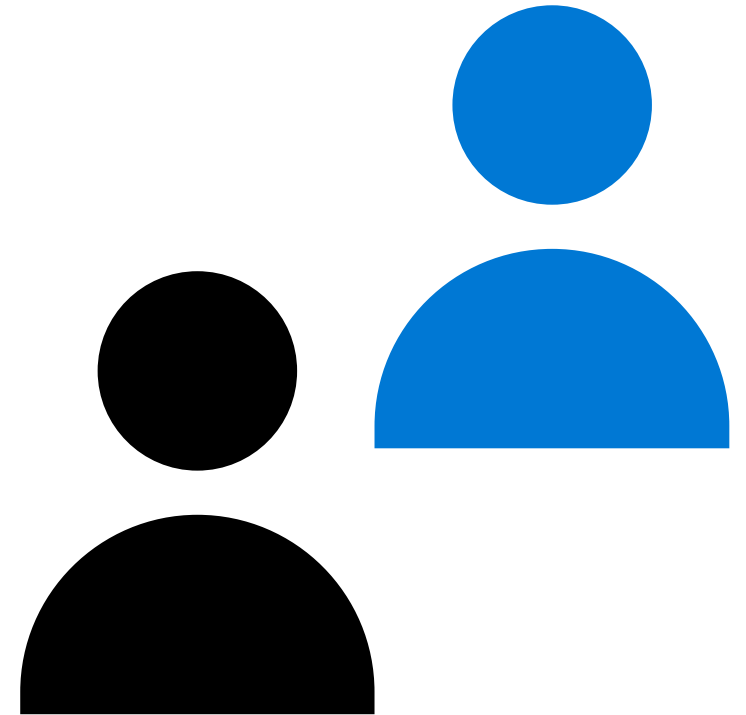
What is your target environment?

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?

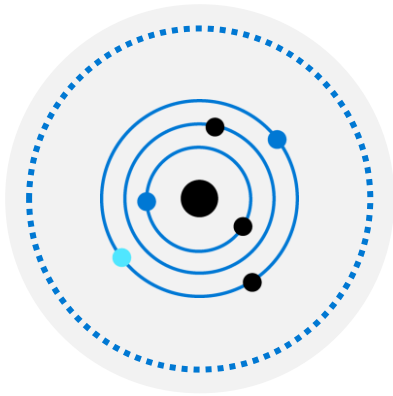
Is there downtime? What happens to performance?
Are users impacted?



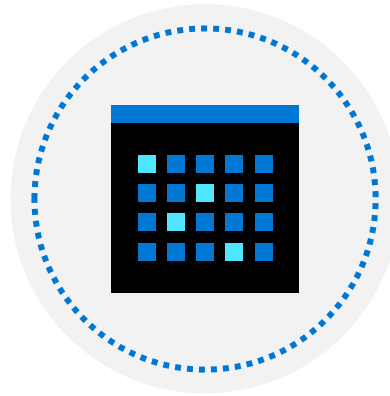
Setting up deployment stages

DEMO

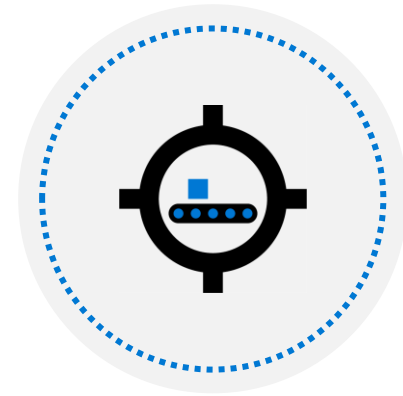
Delivery cadence – three types of triggers



Continuous
deployment trigger



Scheduled trigger



Manual trigger

Selecting your delivery and deployment cadence

DEMO

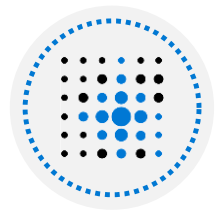
Release approvals



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



Release gate approvals 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 perform validation with other automated systems until specific requirements are verified.

Demonstration: setting up manual approvals

DEMO

Release gates

Release gates give you additional control over the start and completion of the deployment pipeline. They are often set up as a pre-deployment and post-deployment conditions.



Incident and issues management



Notification of users by integration with collaboration systems



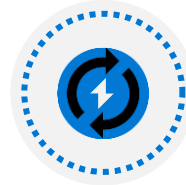
Quality validation



Security scan on artifacts



User experience relative to baseline



Change management

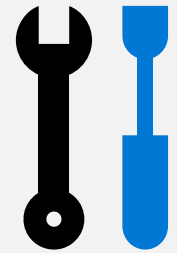


Infrastructure health

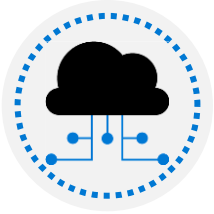
Setting up a release gate

DEMO

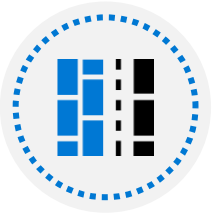
Lesson 04: Building a high-quality release pipeline



Release process (aka 'Release pipeline') versus release



The **release process** involves all the steps that you go through when you move your artifact that comes from one of the artifact sources.



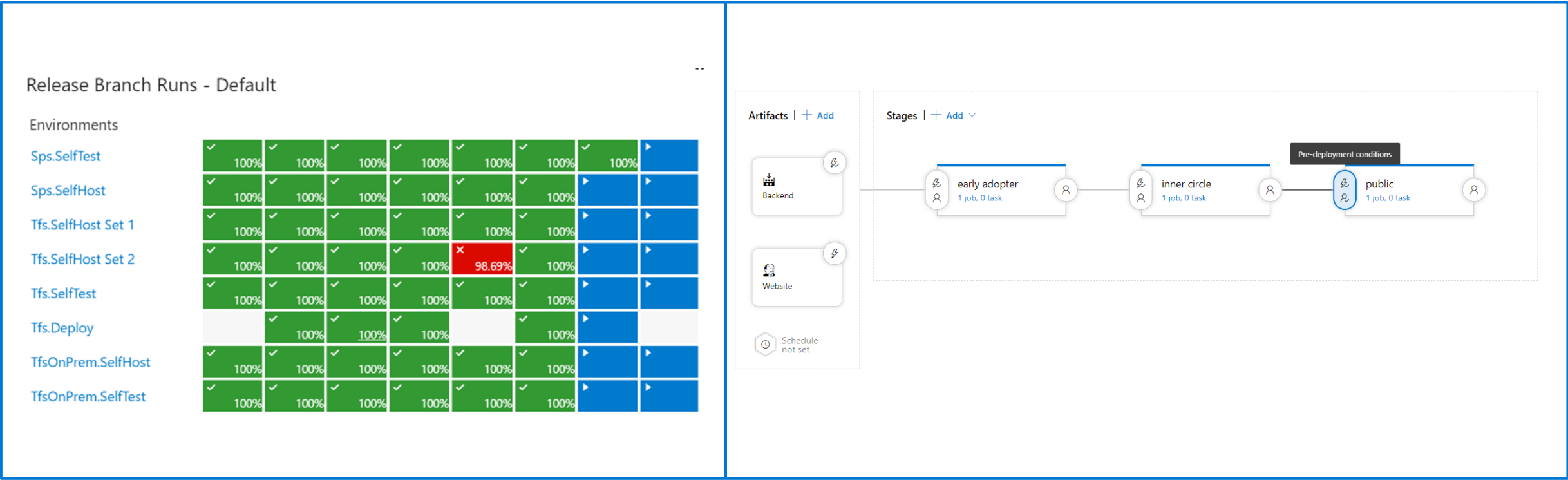
A **release** is a package or container that holds a versioned set of artifacts specified in a release pipeline in your CI/CD process. It holds all the information required to carry out all the tasks and actions in the release pipeline, such as the stages (or environments), the tasks for each one, values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

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



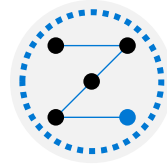
Using release gates to protect quality



No new blocker issues (e.g. bug count)



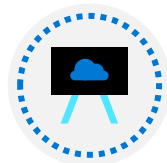
Code coverage on new code greater than 80%



No license violations



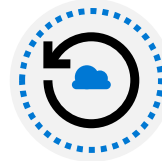
No vulnerabilities in dependencies



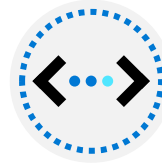
No new technical debt introduced



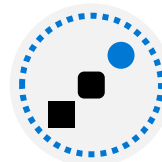
Compliance checks



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

(#mw) This is about where to put the **Release notes** so that the end-users can access them.

Technical or Functional Documentation?

- Where to store Documentation:
- Document Store (e.g SharePoint)
- Wiki (e.g SharePoint, AzDevOps)
- In the code base
- In a Work Item (in AzDevOps)

FEATURE 344

344 Shopping Cart should be personalized

Unassigned0 commentsAdd tag

SaveFollowRefreshUndoMore

Updated by rvanosnabrugge just now

Details

StateNewReasonNew featureAreaTest demoIterationTest demo

DescriptionThe shopping cart should know the user

Acceptance CriteriaClick to add Acceptance Criteria

Release NotesHere can be the commercial release notes.

DiscussionAdd a comment. Use # to link a work item, ! to link a pull request, or @ to mention a person.

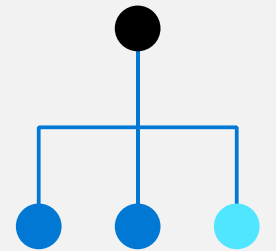
StatusStart DateTarget Date

DetailsPriority2EffortBusiness ValueTime CriticalityValue areaBusiness

DevelopmentAdd linkDevelopment hasn't started on this item.

Related WorkAdd linkThere are no links in this group.

Lesson 05: Choosing the right release management tool



Considerations for choosing release management tools

Artifacts and artifact source

Stages

Triggers and schedules

Build and release tasks

Approvals and gates

Traceability, auditability and security

(#mw) Look at all the considerations in the skillpipe notes!

Common release management tools

Jenkins



Circle CI



Azure DevOps Pipelines



GitLab Pipelines



Atlassian Bamboo

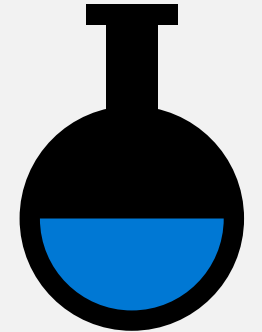


XL Deploy/XL Release



Lesson 06: Labs

Only 10a
Can do 10b outside class time



Controlling deployments using Release Gates

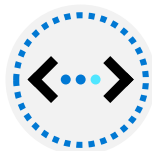


Lab overview:

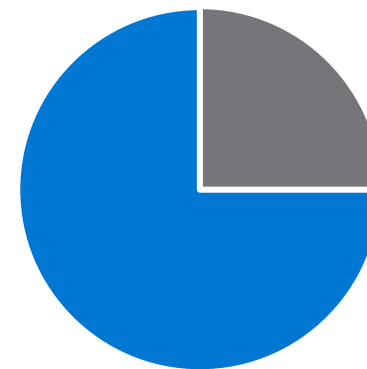
This lab covers the configuration of the deployment gates and details how to use them to control execution of Azure pipelines.

Objectives:

- Configure release pipelines
- Configure release gates
- Test release gates



Duration:



Creating a release dashboard

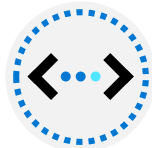


Lab overview:

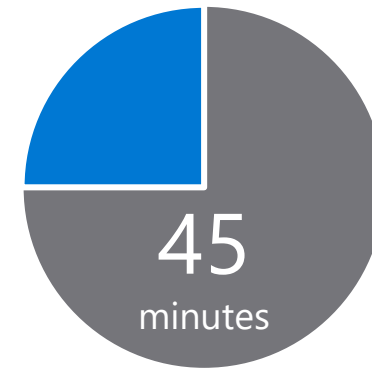
In this lab, you will step through creation of a release dashboard and the use of REST API to retrieve Azure DevOps release data, which you can make this way available to your custom applications or dashboards.

Objectives:

- Create a release dashboard
- Use REST API to query release information



Duration:

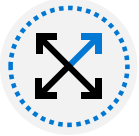


We're skipping this.
Can do outside class time

Lesson 07: Module review and takeaways



What did you learn?



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



Choose a release management tool

Module review questions

1

When you want to change an immutable object of any type, what do you do?

2

What can you use to prevent a deployment in Azure DevOps when a security testing tool finds a compliance problem?

3

Even if you create exactly what a user requested at the start of the project, the solution will often be unsuitable for the same user. Why?