Microsoft

WorkshopPLUS: DevOps Fundamentals

Module 3: Release

Conditions and Terms of Use

Microsoft Confidential

This training package is proprietary and confidential and is intended only for uses described in the training materials. Content and software is provided to you under a Non-Disclosure Agreement and cannot be distributed. Copying or disclosing all or any portion of the content and/or software included in such packages is strictly prohibited.

The contents of this package are for informational and training purposes only and are provided "as is" without warranty of any kind, whether express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Training package content, including URLs and other Internet website references, is subject to change without notice. Because Microsoft must respond to changing market conditions, the content should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

Copyright and Trademarks

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

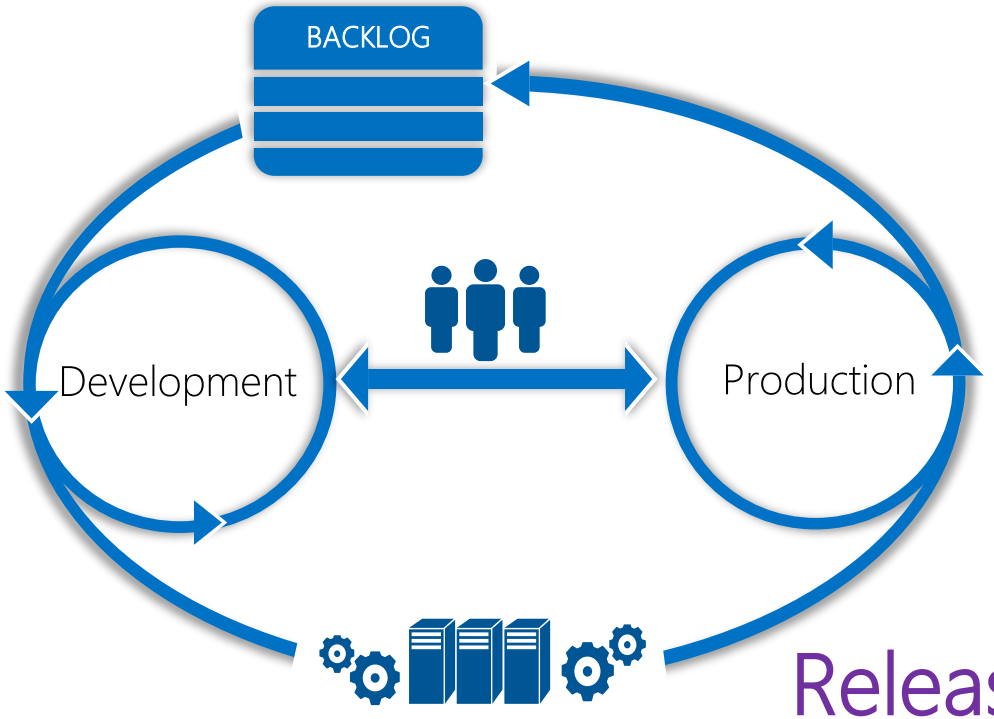
Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

For more information, see **Use of Microsoft Copyrighted Content** at
https://www.microsoft.com/en-us/legal/intellectualproperty/permissions/default.aspx

Microsoft®, Internet Explorer®, Outlook®, SkyDrive®, Windows Vista®, Zune®, Xbox 360®, DirectX®, Windows Server® and Windows® are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other Microsoft products mentioned herein may be either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are property of their respective owners.

Module Objectives

- Understand the importance of configuration management
- Use Infrastructure as Code to standardize configuration management
- Learn how to incorporate and automate all the quality gates of your release process
- See how you can achieve safe continuous delivery



Software Configuration Management

What is SCM (Software Configuration Management)?

- A process to handle changes in software projects
- Used with a configuration database, or CMDB
- Identify the change for the software (physical or functional)
- Implement the change and assess the result of the change
- Provides traceability and higher software integrity

Importance of Software Configuration Management

- Ensure the current design and build state is trusted
- Visibility of changes in the system that can be baselined or compared to an earlier baseline
- Accountability of changes and easy identification of failing change in a production system

Cloud-First World

"scale x complexity exceeds skillset"

Jeffrey Snover

Idempotence

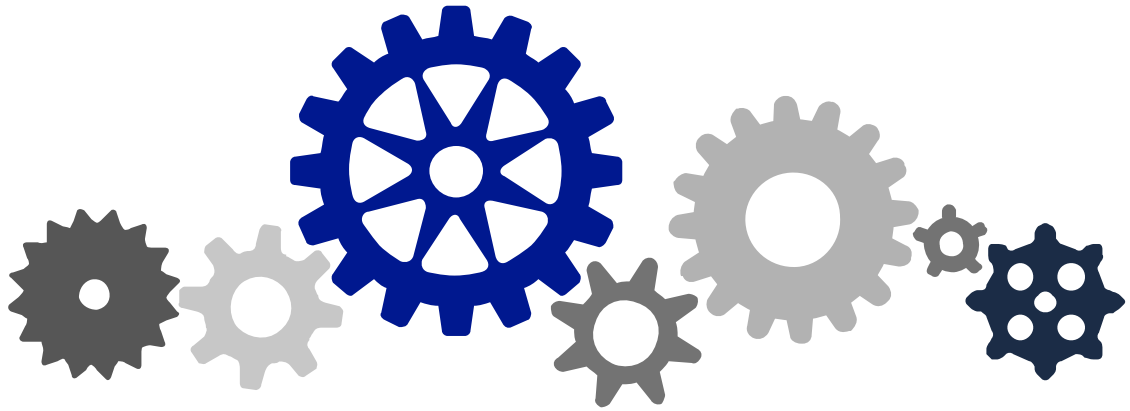
- The property of certain operations in mathematics and computer science, that can be applied multiple times without changing the result beyond the initial application
- In other words, pressing the on button turns it on, whether it is on or off

On/*off* buttons of a Polish desk calculator. Pressing the *On* button (green) is an idempotent operation, since it has the same effect whether done once or multiple times.

https://en.wikipedia.org/wiki/Idempotence

Infrastructure as Code (IaC)

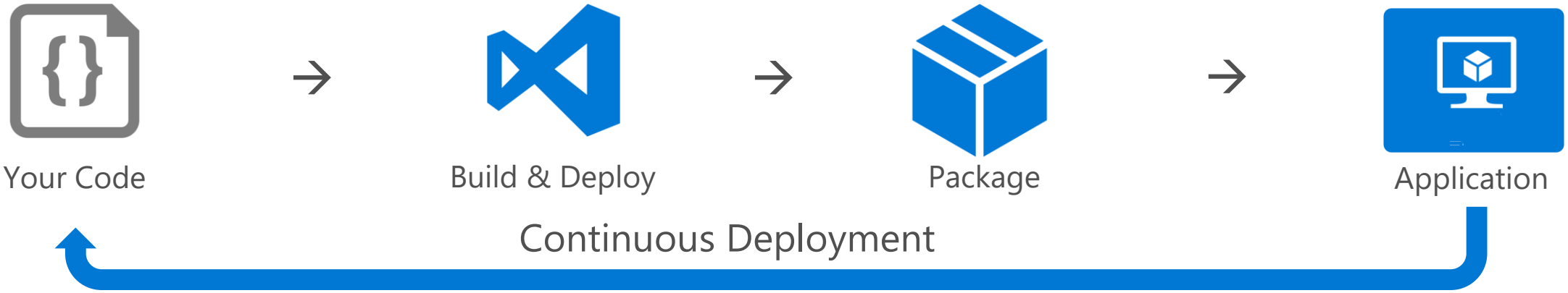
- Infrastructure as Code (IaC) is the management of infrastructure (networks, virtual machines, load balancers, and connection topology) in a descriptive model, using the same versioning as source code.
- An IaC model generates the same environment every time it is applied.



Configuration Management & Continuous Deployment

Configuration Management

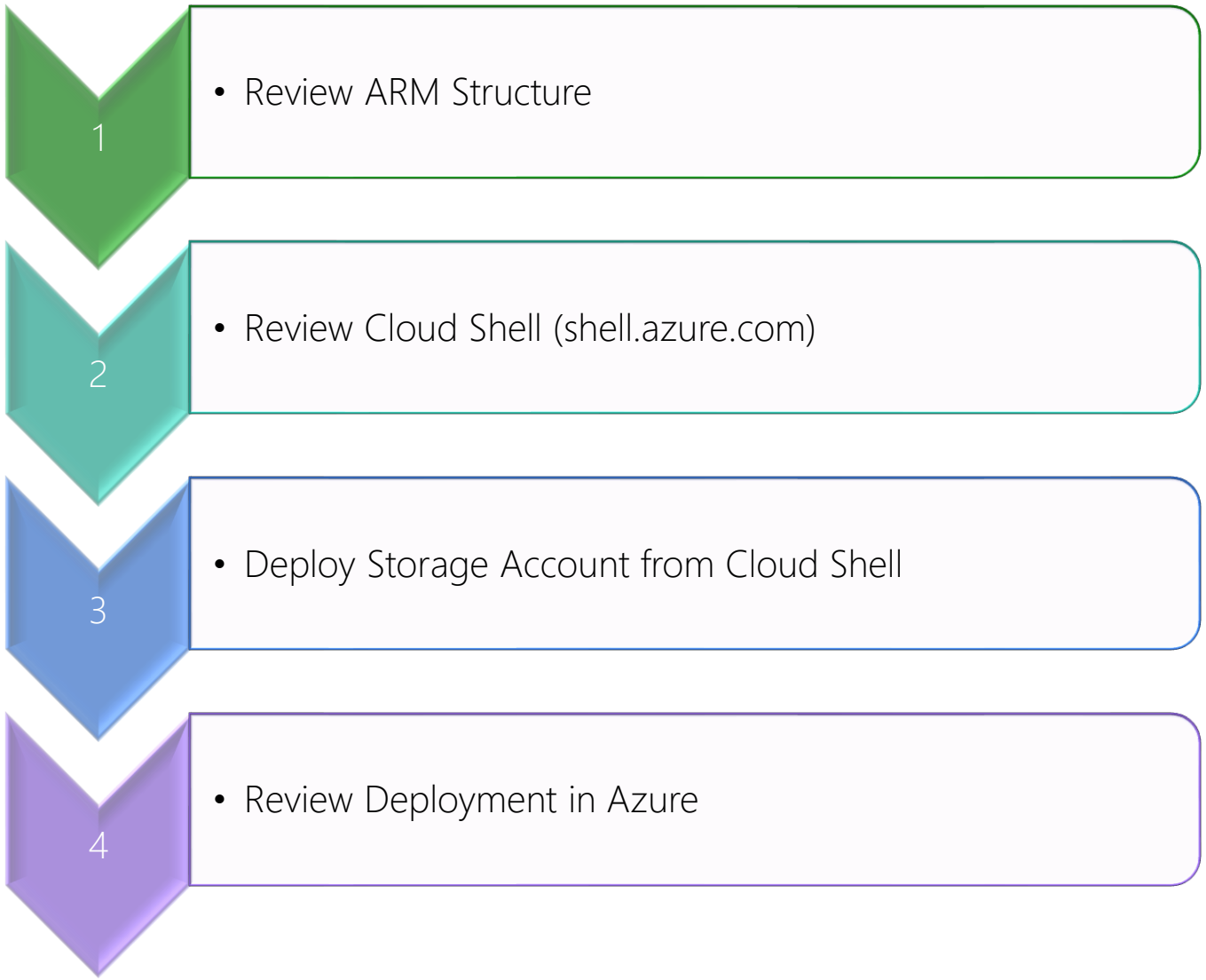
Deployment



Demonstration:
Infrastructure as Code

We will use an Azure Resource Manager template to define and deploy a storage account to Azure.

Demonstration Review



Continuous Integration

What is Continuous Integration?

"… a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly."

Continuous Integration. (2006). martinfowler.com. https://martinfowler.com/articles/continuousIntegration.html

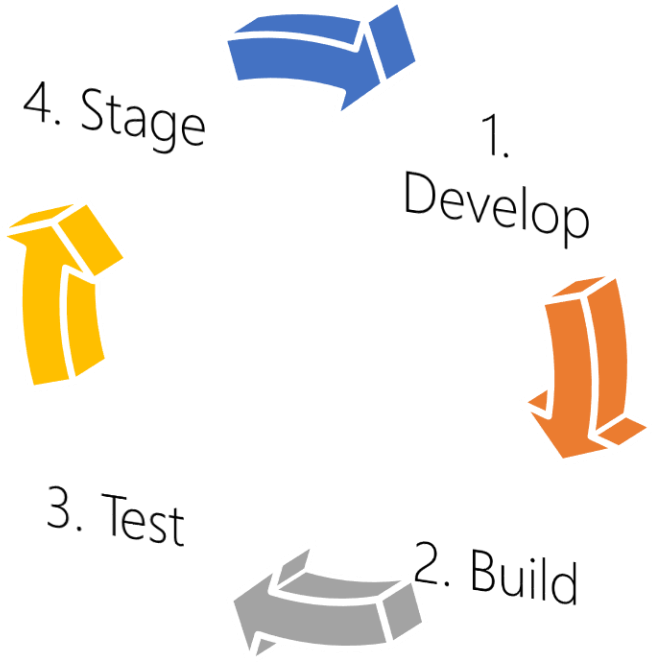
Continuous Integration Practices

- Maintain a single source repository
- Automate the build
- Everyone commits to the mainline every day
- Every commit should build the mainline
- Fix broken builds immediately
- Test in a clone of production environment
- Automate deployment

Continuous Integration. (2006). martinfowler.com. https://martinfowler.com/articles/continuousIntegration.html

Cyclical Processes found in Continuous Integration

1. Develop
   - Use TDD and create Unit Tests
   - Use Built in Quality which is a set of practices putting quality first
   - Use Built in Security which is a set of practices putting security first
   - Decompose Features to Stories
   - Use Version Control Practices
2. Build
   - Automatic triggers
   - Work out of main branch
3. Test (end-to-end)
   - Use Infrastructure as Code
4. Stage
   - Deployment Rings
   - Monitor and Review Feedback



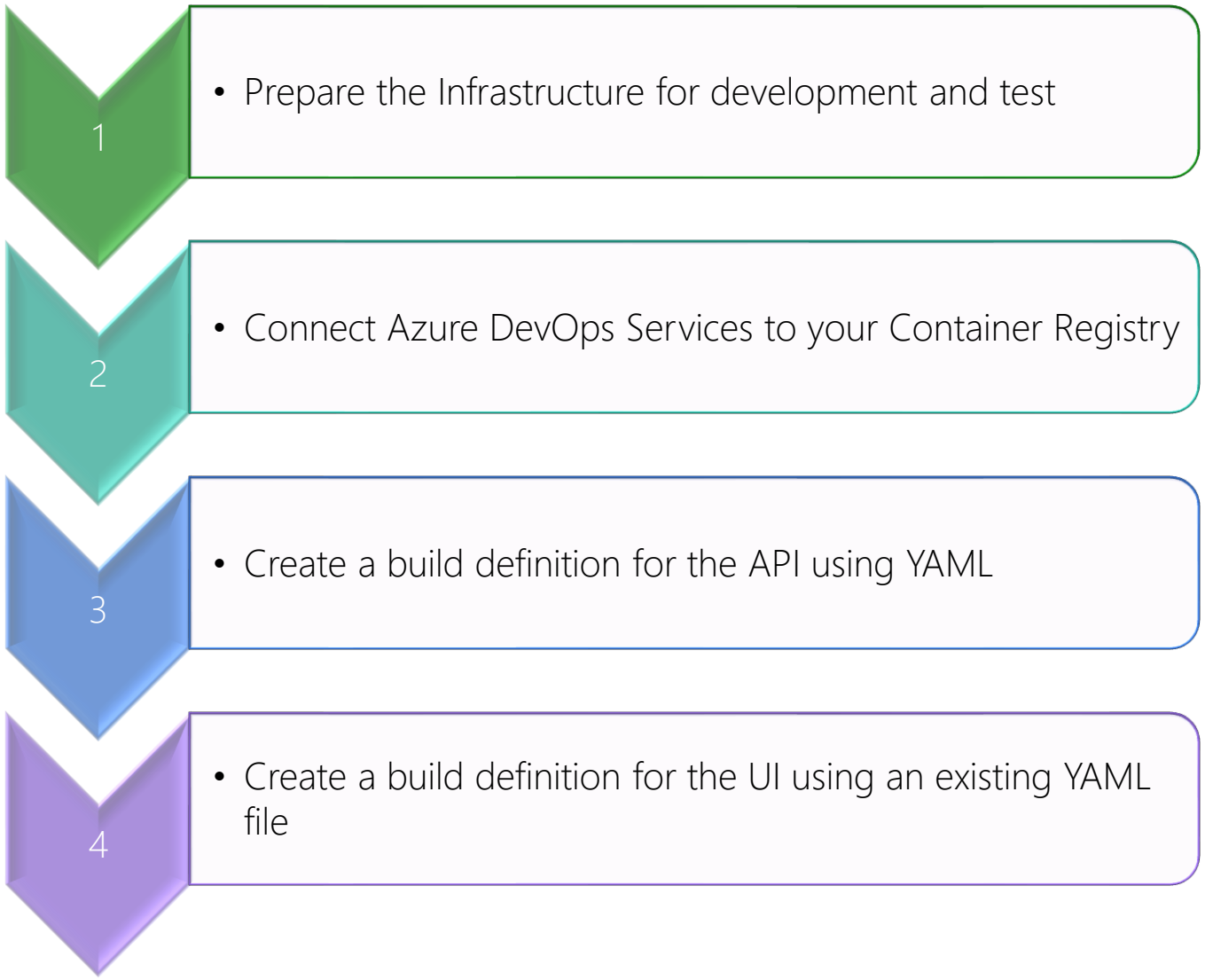
Strategy, Tips, and Practices

- Use Branch Policies
- Don’t be afraid to implement continuous integration where automation means builds fire on merging to main
- Create a team process, for example topic branching
- Units tests are imperative, test what is appropriate, and develop a taxonomy for your test suite
- Security is imperative; bring security in early
- Don’t forget telemetry (*more of this in module 4*)

Demonstration: Continuous Integration

We will use Azure DevOps Services to promote successful builds through various states and get feedback.

Demonstration Review



Module 3: Release

Lab 5: Continuous Integration

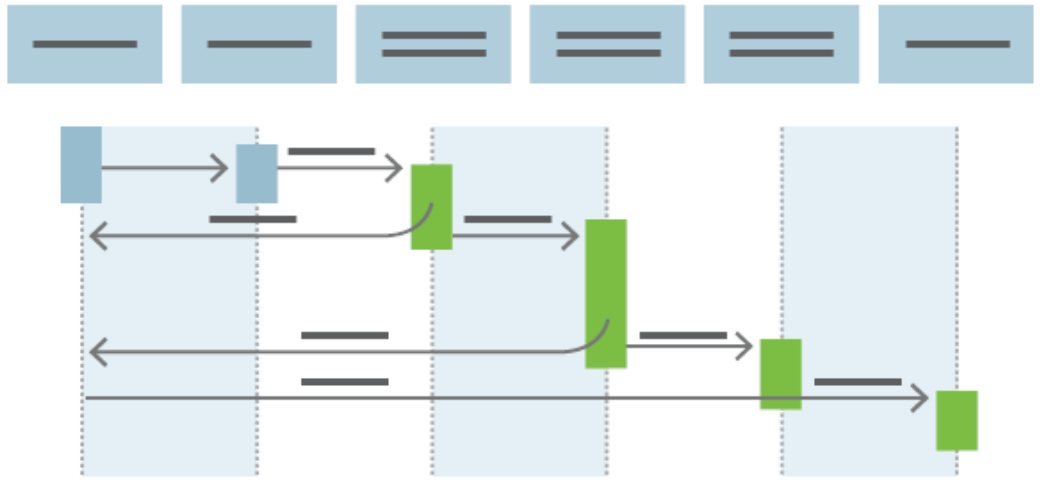
Exercise 1: Prepare the Infrastructure
Exercise 2: Connect Azure DevOps to your ACR
Exercise 3: Create a Build Definition for the API
Exercise 4: Create a Build Definition for the UI

Lab Time: 60 minutes (about 1 hour)

Continuous Delivery

What is Continuous Delivery?

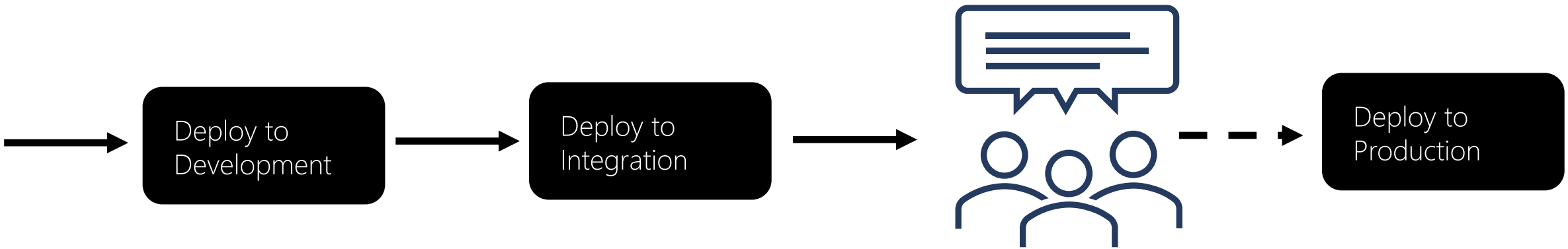
- Software development discipline where you build software so that it can be released to production at any time
- You achieve continuous delivery by continuously integrating software changes, building executables or other binaries, and running automated tests on those binaries to detect problems



Continuous Delivery. (2013). martinfowler.com. https://martinfowler.com/articles/continuousdelivery.html

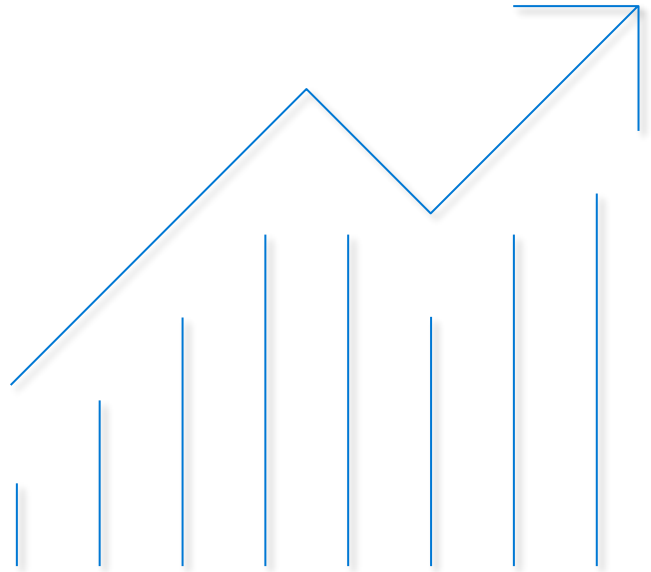
Continuous Delivery vs Continuous Deployment

- Continuous Deployment sees every change through a pipeline put into production with many production deployments every day
- Continuous Delivery means you can do frequent deployments but may choose not to, usually as a result of the business needing a pause



Continuous Delivery Goals

- Low risk releases (typically incremental)
- Faster time to market
- Higher quality
- Lower costs
- Better products
- Happier teams (better morale)
- Automate (minimizes TTD (time to deploy), TTM (time to mitigate), KPIs)



Continuous Delivery. (2017). continuousdelivery.com. https://continuousdelivery.com

Continuous Delivery Methods

- Sequential Rings

Stage code and deploy to "pockets" of reviewers. Often referred to as controlling the blast radius

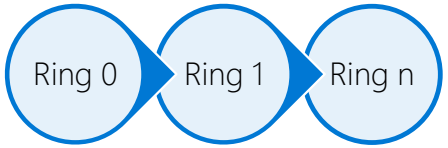
- Blue / Green Deployments

Switch server with a load balancer, or traffic manager. In case of failure switch the servers back (seamless)

- Feature Flags

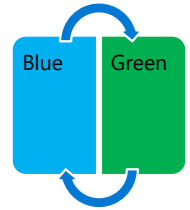
Control what can be used, and turn off or on features when applicable

Sequential Rings



- Continuous Delivery may sequence multiple deployment "rings" for progressive exposure
- Progressive exposure groups users who get to try new releases to monitor their experience in "rings"
- The first deployment ring is often a "canary" used to test new versions in production before a broader rollout
- CD automates deployment from one ring to the next and may depend on an approval step, in which a decision maker signs off on the changes electronically

Blue / Green Deployment



- “Blue/Green deployment” relies on keeping an existing (blue) version live while a new (green) one is deployed
- Uses load balancing to direct increasing amounts of traffic to the green deployment
- If monitoring discovers an incident, traffic can be rerouted to the blue deployment still running

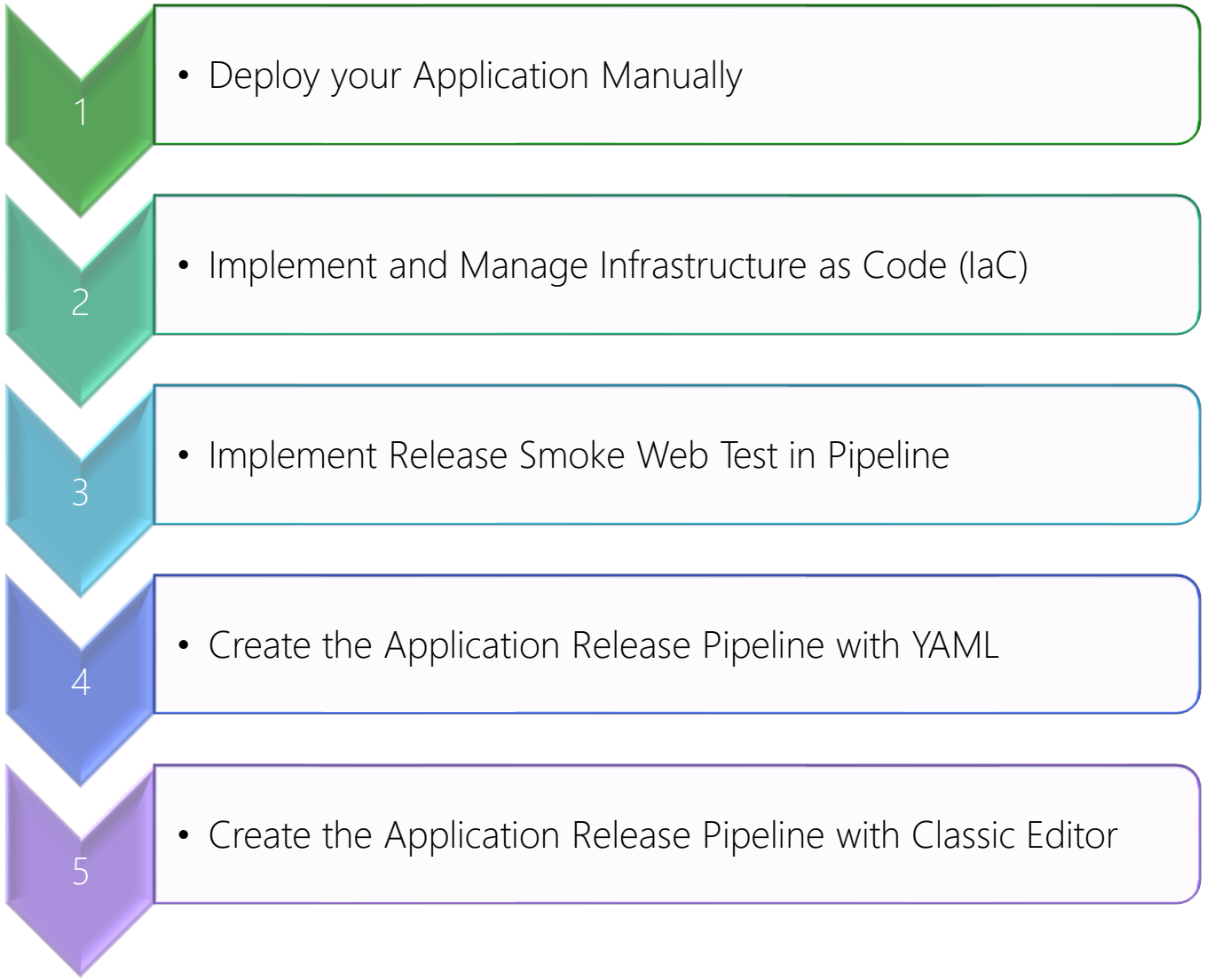
Feature Flags

- “Feature flags” comprise another technique used for experimentation and “dark launches”.
- Feature flags turn features on or off for different end users based on their identity and/or group membership.

Demonstration: Continuous Delivery using YAML and Classic Pipeline

We will use Azure DevOps Services to deploy through successful builds and implement ring deployment methodology.

Demonstration Review



Module 3: Release

Lab 6: Continuous Delivery

Exercise 1: Deploy your Application Manually
Exercise 2: Manage Infrastructure as Code
Exercise 3: Install Release Smoke Web Test
Exercise 4: Create the Release Pipeline (yaml)

Lab Time: 60 minutes (about 1 hour)

Knowledge Check

Question #1: What is Configuration Management?

A process to handle changes in software projects, usually managed with a CMDB (configuration management database), or a CMS (configuration management software) suite.

Question #2: What is Infrastructure as Code?

The management of infrastructure (networks, virtual machines, load balancers, and connection topology) in a descriptive model using the same versioning as source code.

Question #3: What is Release Management?

The process of managing, planning, scheduling and controlling a software build through different stages and environments.