European
Power Platform
Conference

BRUSSELS
11–13 JUNE 24

From Zero to ALM

Benedikt Bergmann

CEO, CRM Konsulterna

Benedikt Bergmann

https://bio.link/benediktbergmann

CEO @CRM Konsulterna DE

Power Platform Consultant

@CRM-Konsulterna in Stockholm

- .Net development
- Angular
- Dynamics CRM/CE/MDA
- Power Platform

Twitter: https://twitter.com/BergmannBene

LinkedIn: https://www.linkedin.com/in/benedikt-bergmann

Mail: benedikt@benediktbergmann.eu

Blog: http://benediktbergmann.eu

Questions

Agenda

- What is ALM?
- Basics for Power Platform
  - Solutions
  - Environments
  - Power Platform Pipelines vs. ALM Accelerator vs. ADO Pipeline
- Basic ALM
- Azure DevOps Pipelines & GitHub Actions
  - YAML
  - Build Tools
- Environment Variables
- Connection References
- “Advanced” YAML
- Versioning
- Quality Gates
  - Automated Tests
  - Solution Checker
  - …
- Run PAC CLI in pipeline
- Move Data & Portal Configuration
- Project setup
- Pull Requests
- Other ALM approaches
  - Use Build Env
  - Pack from source control
  - Branching

Time Schedule / Breaks

- 10:30 AM to 11:00 AM: Morning Break
- 12:45 PM to 2:00 PM: Lunch
- 3:15 PM to 3:45 PM: Afternoon Break
- 5:00 PM (ish): Conclusion & End

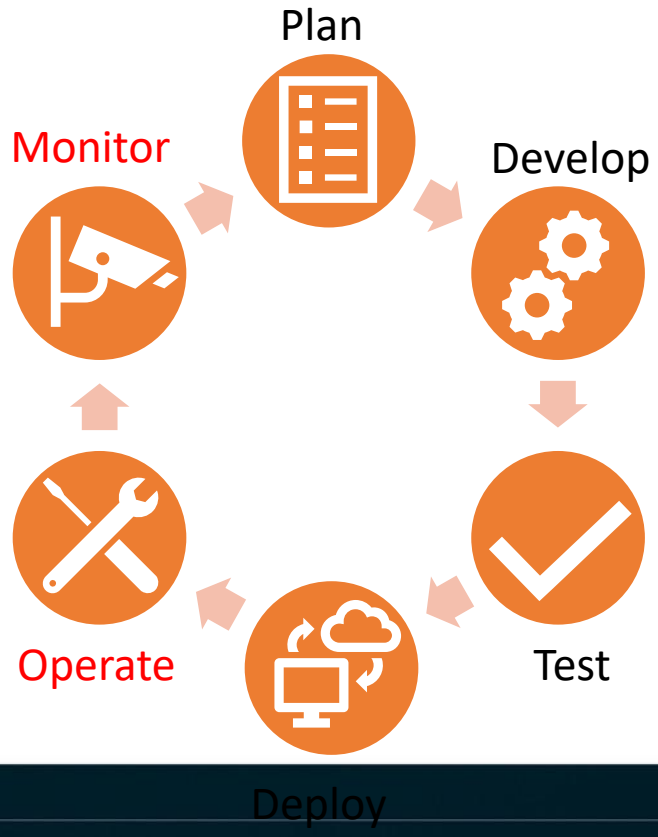
What is ALM?

Application lifecycle management (ALM) is an integrated system of people, tools and processes that supervise a software application from its initial planning and development, through testing and maintenance, and into decommissioning and retirement.

TechTarget

What is ALM?

- Application Lifecycle Management
- Automation



Why ALM?

- Increased visibility into workflow
- Enhanced compliance
- Faster deployments
- Higher-quality products

- Automation

- Minimize manual work
- Increase Quality

Tools ALM

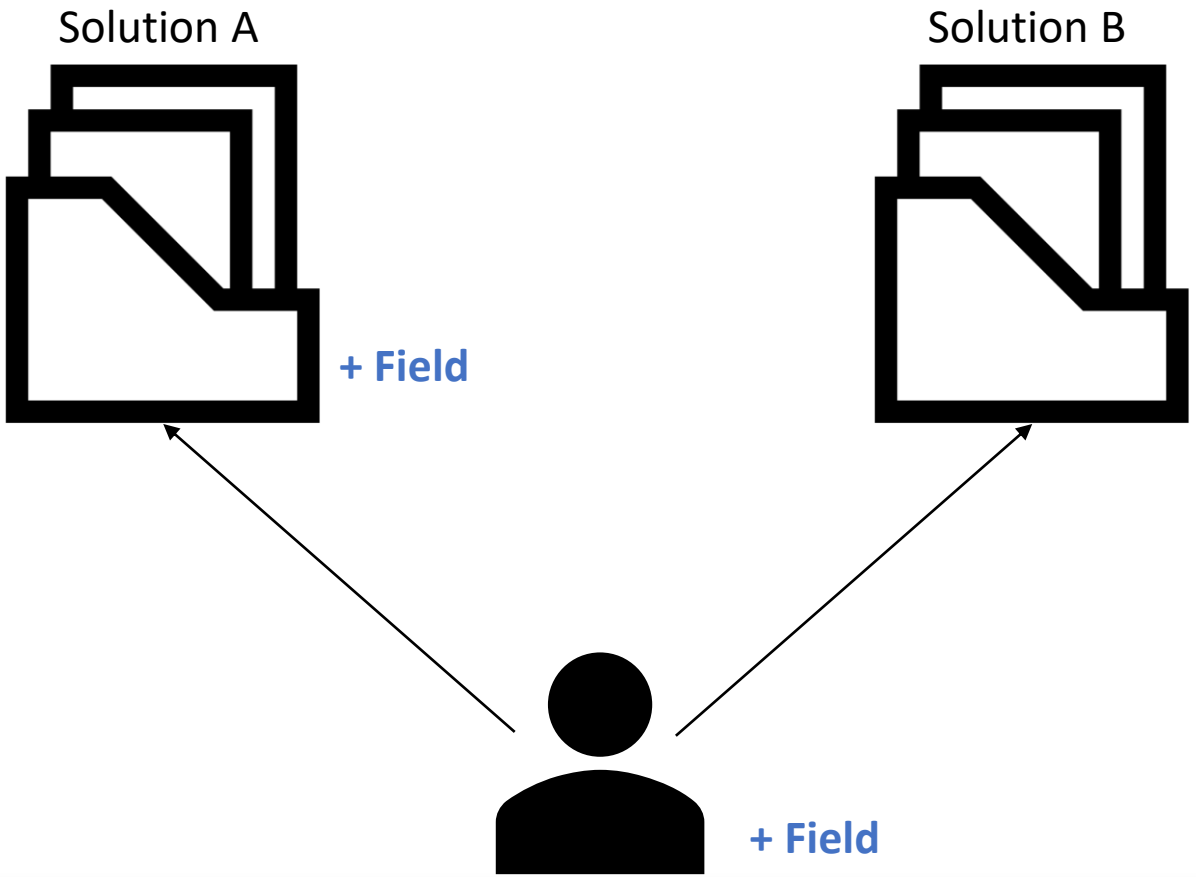
- Version Control
- Team communication
- Estimation and planning
- Requirement management
- Test management
- Maintenance and support
- Automated deployment
- …

Basics for Power Platform

Solutions

- **Only Containers**
- Never use ”Include all components”
- Configurable
- Contains ”only” customization
- Purpose/Supplier separation
  - Note publisher
- Patching & Cloning

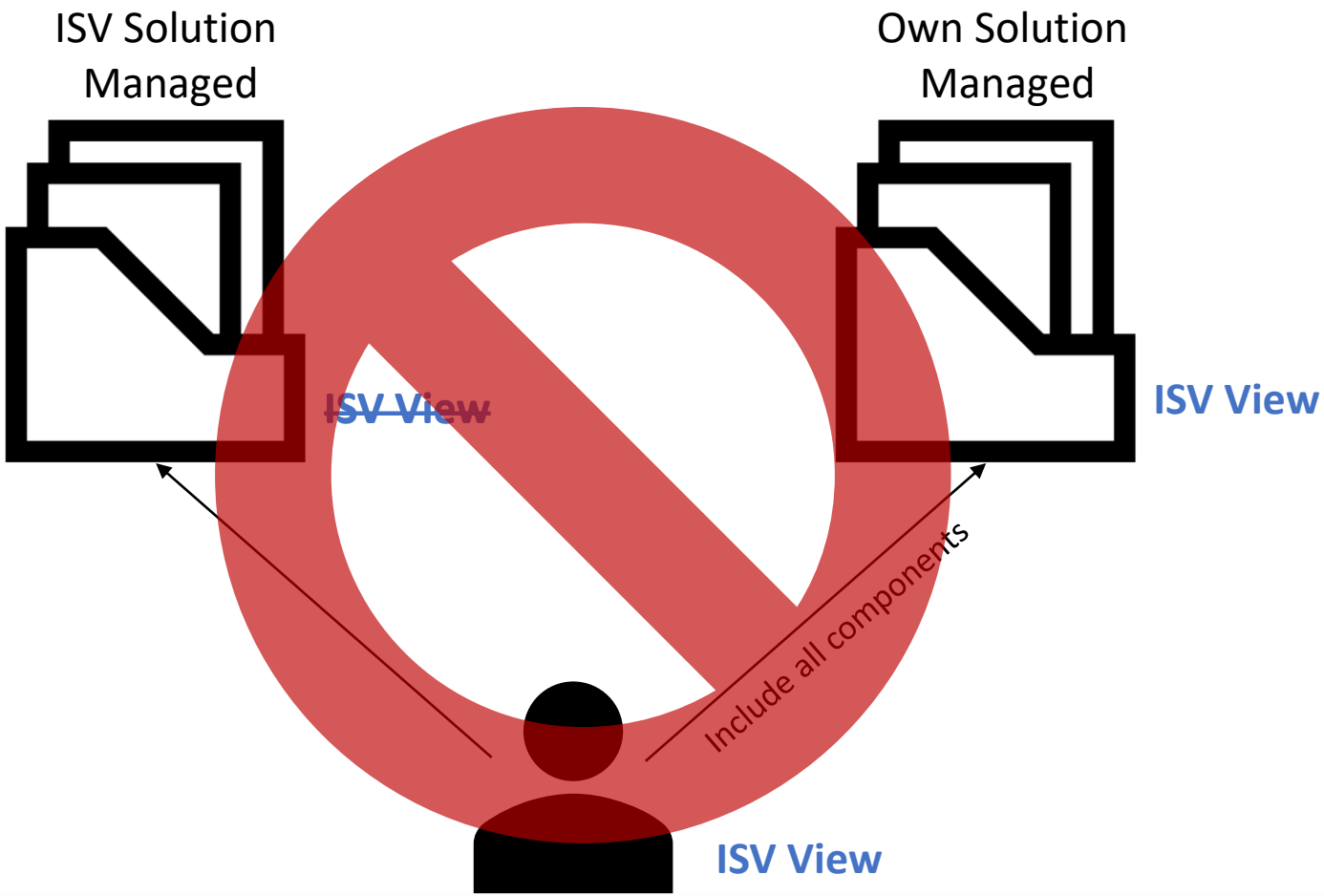
Solutions



Solutions

- Only Containers
- **Never use ”Include all components”**
- Configurable
- Contains ”only” customization
- Purpose/Supplier separation
  - Note publisher
- Patching & Cloning

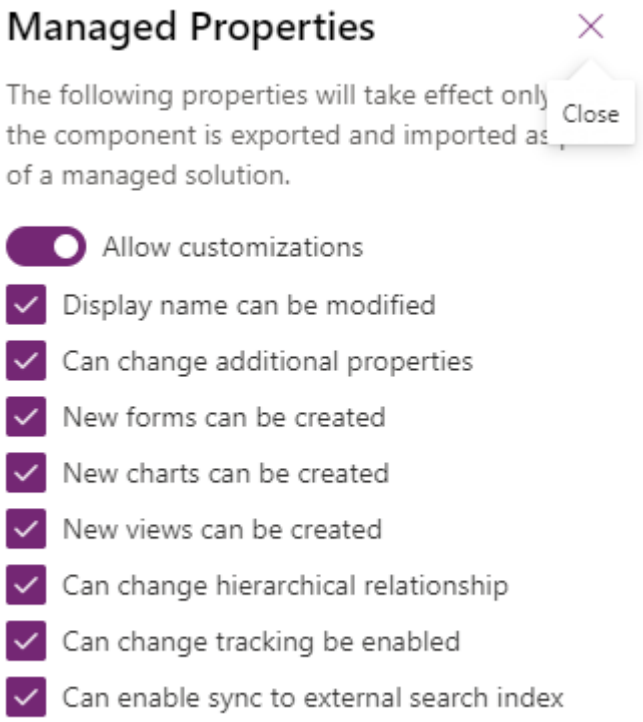
Solutions


ISV Solution
Managed
Own Solution
Managed
ISV View
ISV View
Include all components
ISV View
Contact Entity

Solutions

- Only Containers
- Never use ”Include all components”
- **Configurable**
- Contains ”only” customization
- Purpose/Supplier separation
  - Note publisher
- Patching & Cloning

Solutions


Managed Properties
The following properties will take effect only
the component is exported and imported as
of a managed solution.
Close
Allow customizations
Display name can be modified
Can change additional properties
New forms can be created
New charts can be created
New views can be created
Can change hierarchical relationship
Can change tracking be enabled
Can enable sync to external search index

Solutions

- Only Containers
- Never use ”Include all components”
- Configurable
- **Contains ”only” customization**
- Purpose/Supplier separation
  - Note publisher
- Patching & Cloning

Customization vs. Data

Customization

- Tables
- Fields
- Business Rules
- …

Data

- Table Rows

Exception

- Environment Variables
- Duplication Detection Rules
- …

Solutions

- Only Containers
- Never use ”Include all components”
- Configurable
- Contains ”only” customization
- **Purpose/Supplier separation**
  - Note publisher
- **Patching & Cloning**

Unmanaged vs. Managed

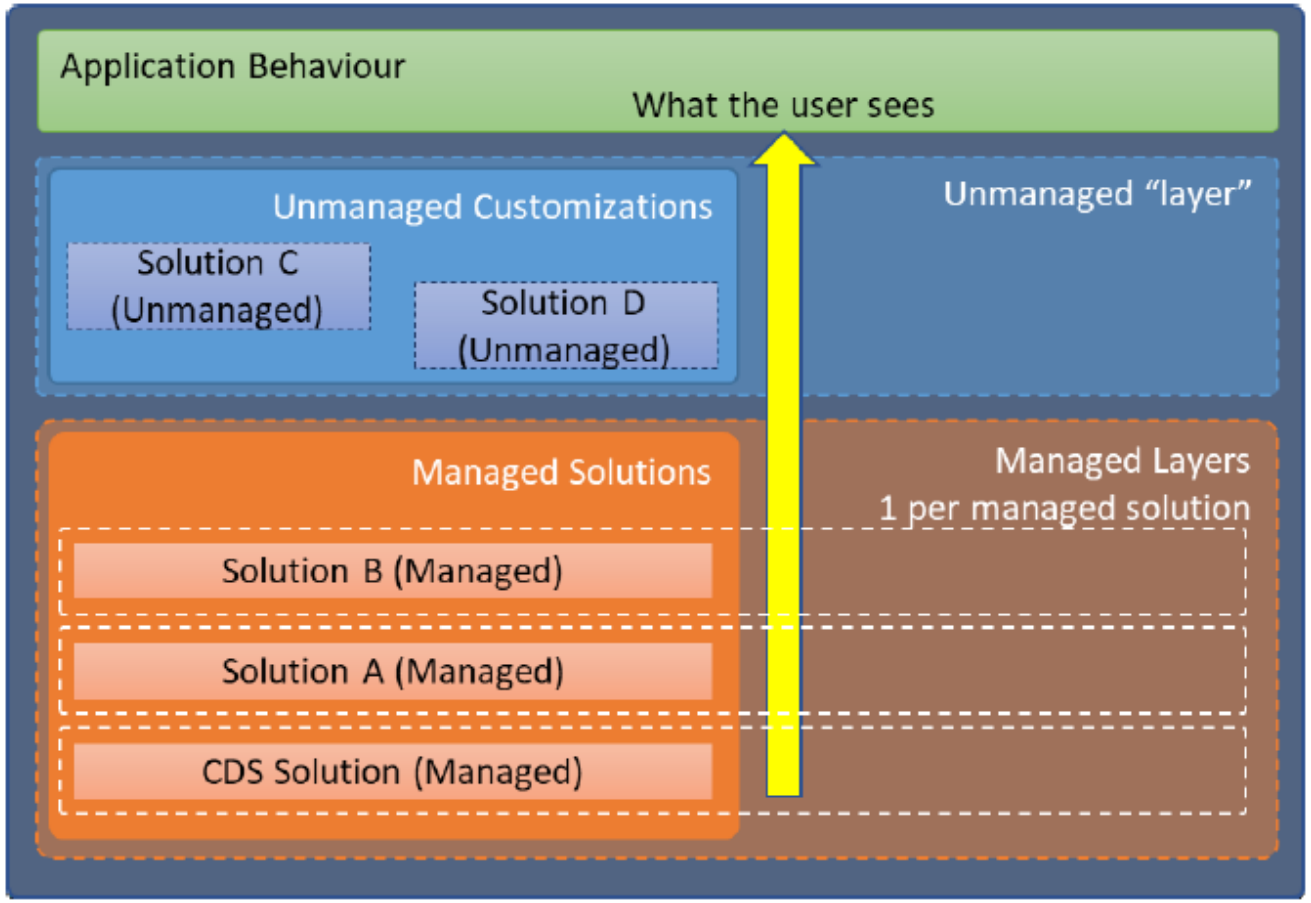
Unmanaged

- Add components
- Remove components
- (Delete components)
- Export to Unmanaged
- Export to Managed
- Can be deleted without uninstalling components

Managed

- Unable to add components
- Unable to remove components
- Can not be exported
- Deleting will uninstall all components

Layering


Application Behaviour
What the user sees
Unmanaged Customizations
Solution C (Unmanaged)
Solution D (Unmanaged)
Unmanaged "layer"
Managed Solutions
Managed Layers
1 per managed solution
Solution B (Managed)
Solution A (Managed)
CDS Solution (Managed)

Solution Structure

- Single Solution
- Multiple Solutions
  - One solution per environment
  - One publisher across solutions
  - Split horizontal or vertical or combination

Solution Structure

- Horizontal

Component of same type

Visual Components	Processes & Plugins	Reports	Security Roles	Main
Apps	Processes	Reports	Security roles	Tables
Canvas components	Plug-in assemblies			Choices
PCF components	SDK message processing steps			Client extensions
Web resources				Service endpoints
				Dashboards
				Connection roles
				Email templates
				Column security profiles

- Vertical

Solution per functional area

Specific sports	Shared base
Basketball	Sports management base
Baseball	

Update vs. Ugrade

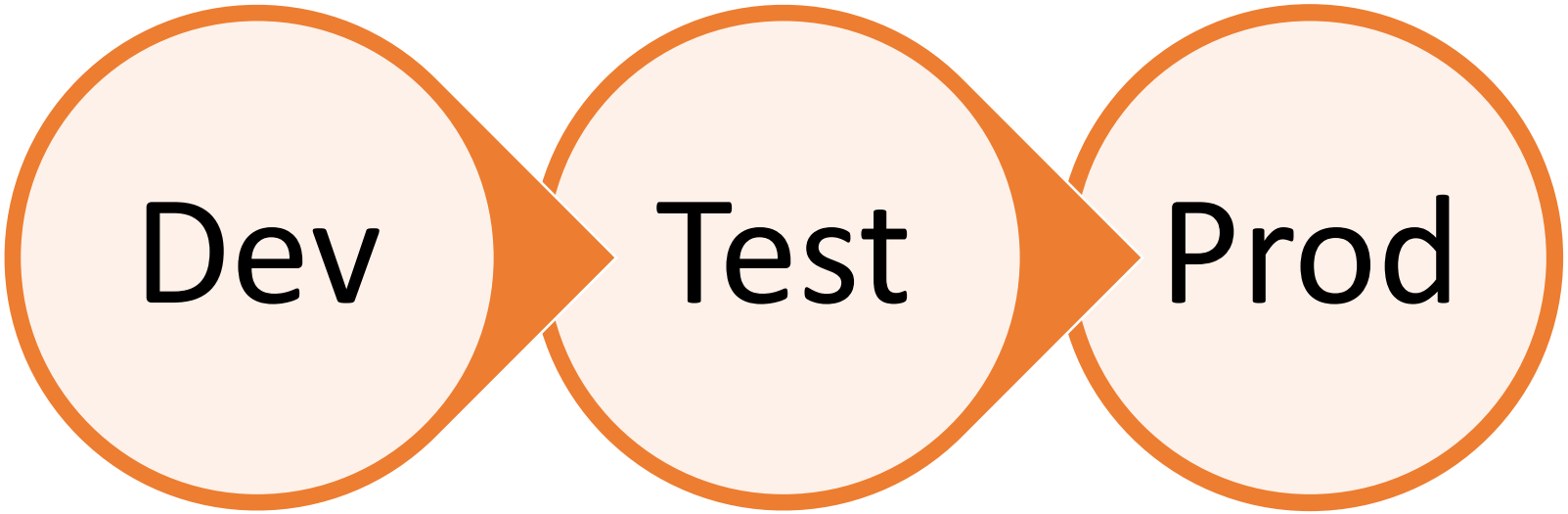
Update

- Installs new components
- Updates existing components

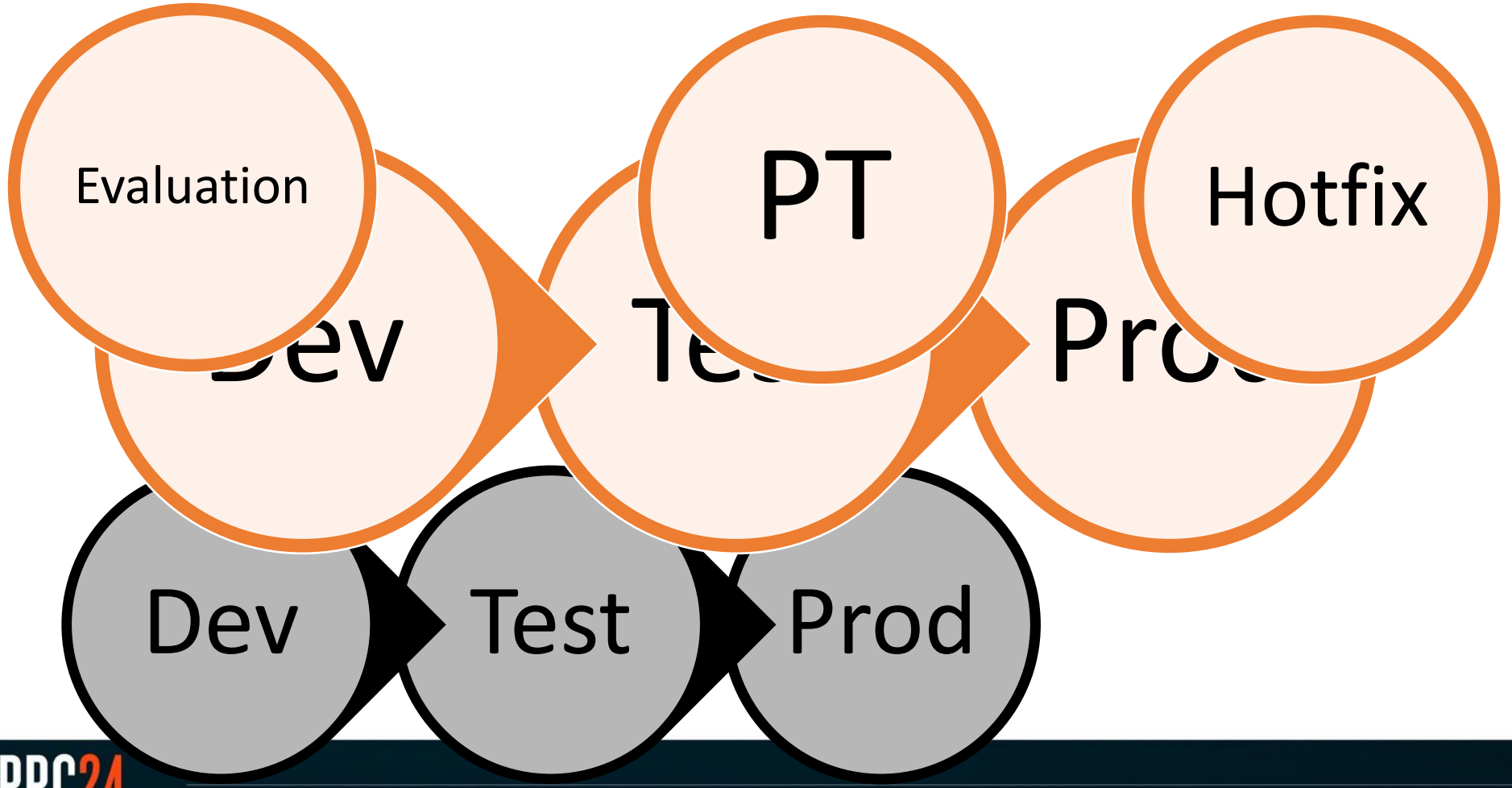
Upgrade

- Installs new components
- Updates existing components
- Not available when same version
- Uninstalls components deleted from the Solution

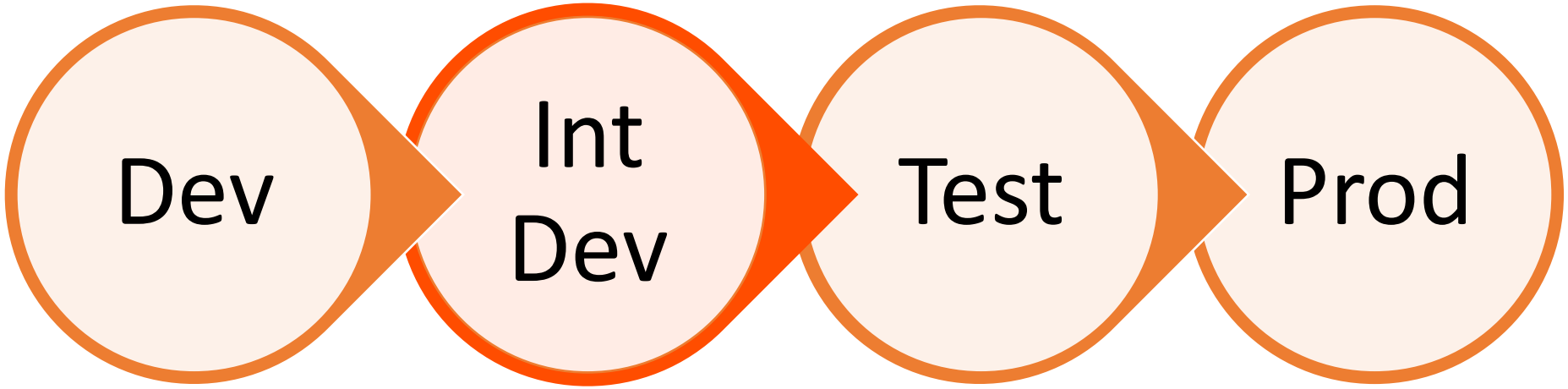
Environments - Minimum



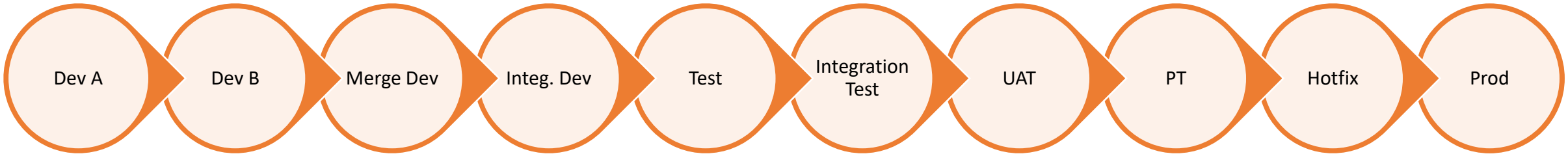
Environments - Sourounding



Environments – Internal dev



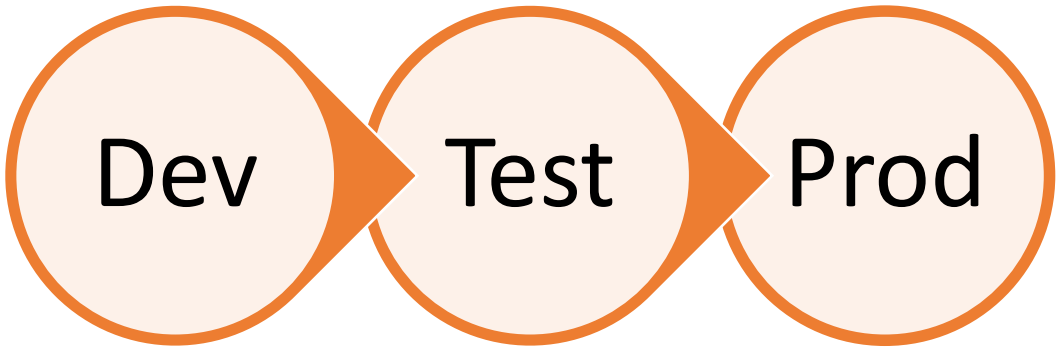
Environments



Environments – Function separation

Sales
Sweden

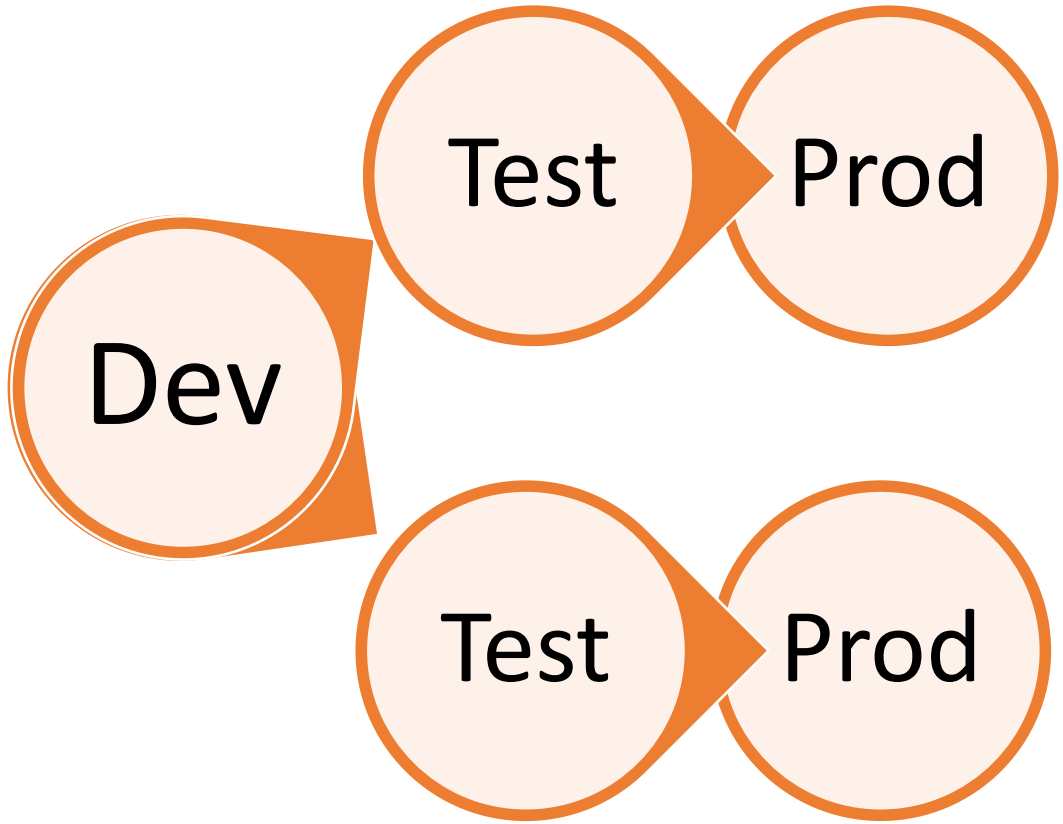
Marketing
Germany



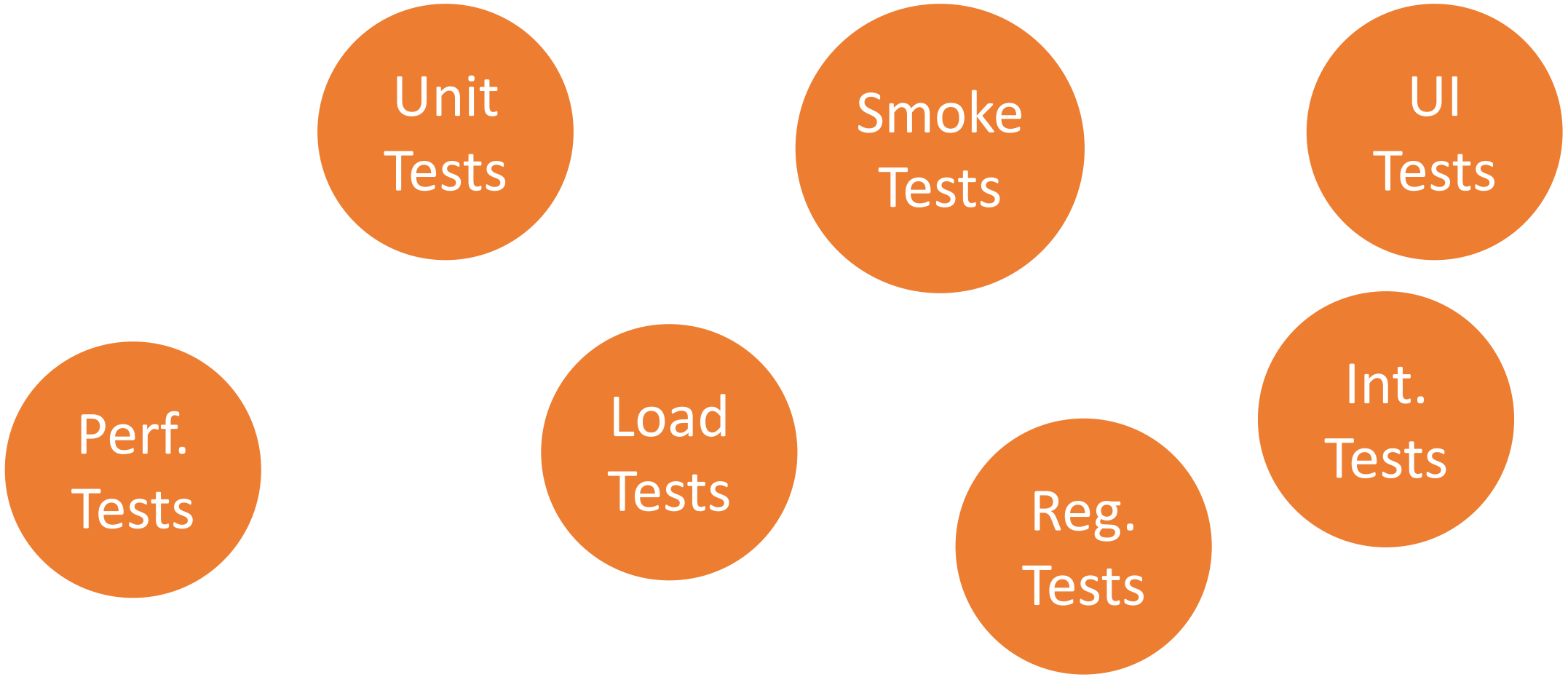
Environments – Data separation

Sales
Sweden

Marketing
Germany

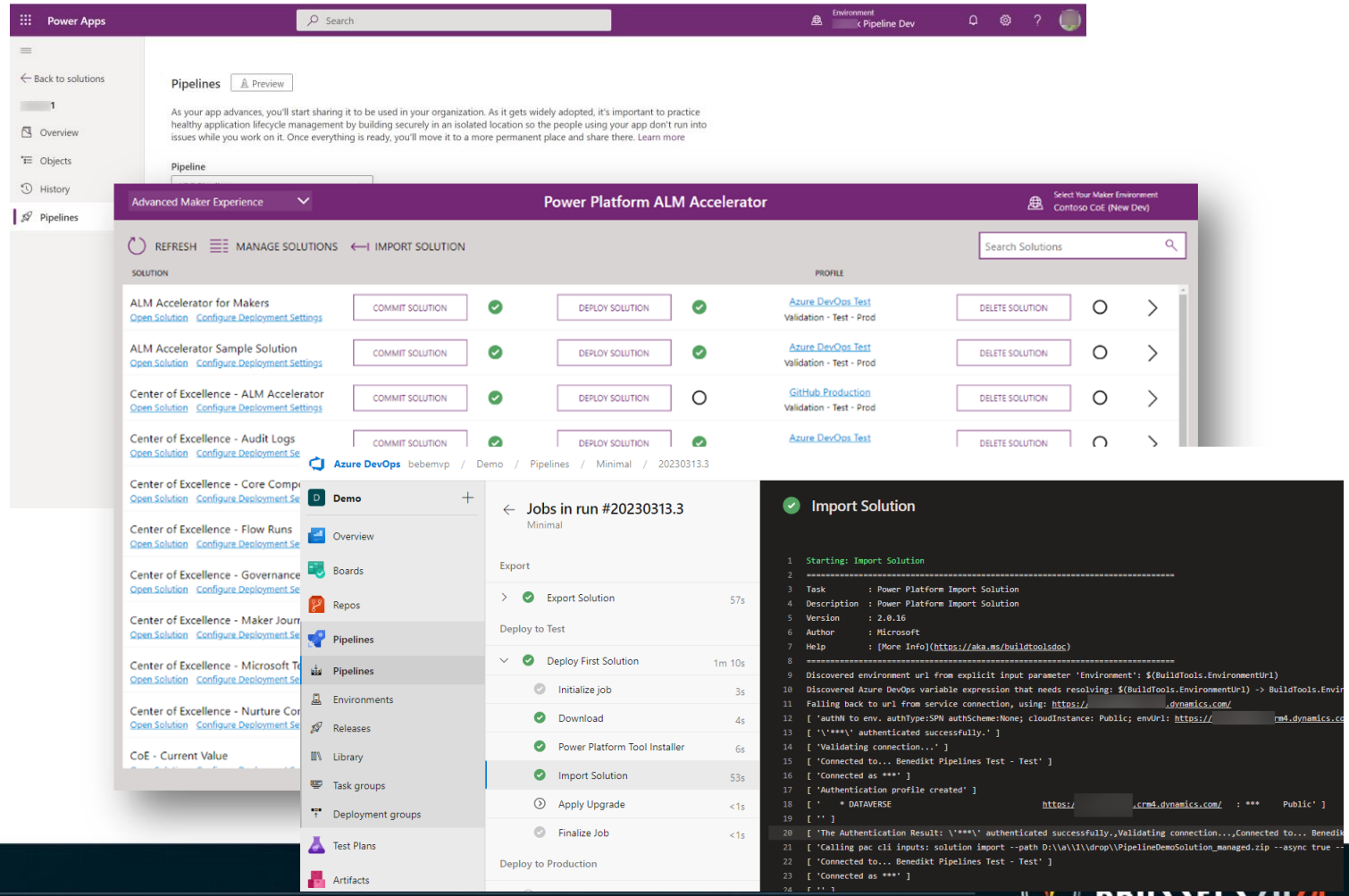


Different Tests



Pipeline ≠ Pipeline

- Power Platform Pipelines
- ALM Accelerator
- Azure DevOps Pipelines



Limitations

- Embedded Canvas Apps
- SLA
- ARC
- Custom Connectors
- Custom Table must be included first
- Managed Plugins
- Service Endpoints

Licensing enforcement

- Owner doesn’t have sufficient license
  - Service Principal
  - Trial
  - No license longer
- Relate to App
  - Automatically
  - Manual (Script or UI)

Basic ALM

Prerequisite

- Azure DevOps project
- Power Platform Build Tools
- Application Registration
- Connections
  - Dev
  - Test

Source code centric vs. environment centric

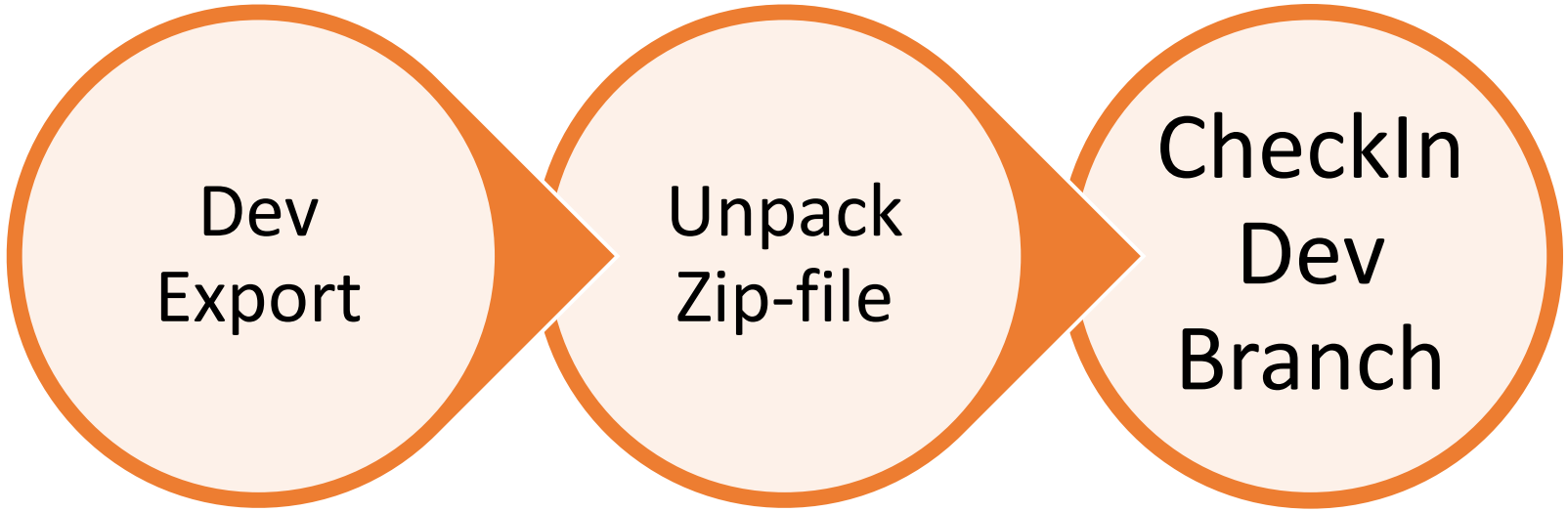
- Source code is truth
- Complex
- More flexible
- Makes branching possible

- Environment is truth
- Easier to setup
- Standard

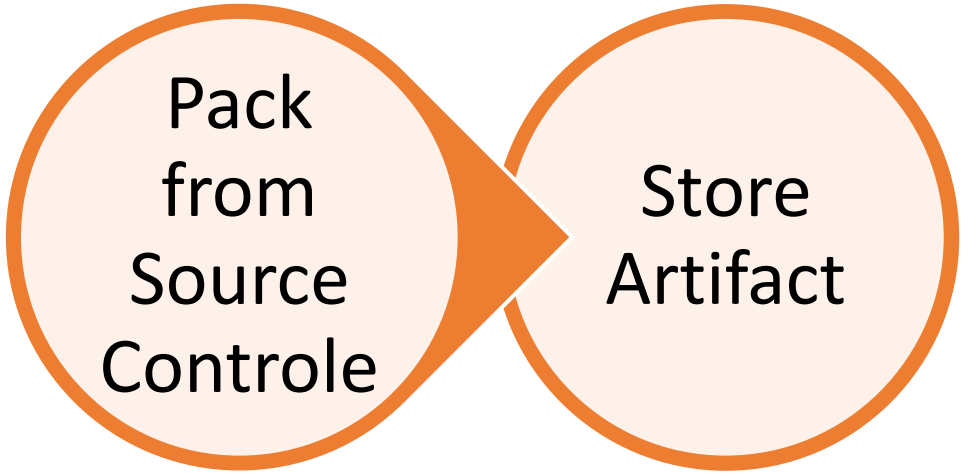
Pipelines

1. Export from Dev
2. Build Solution
3. Release

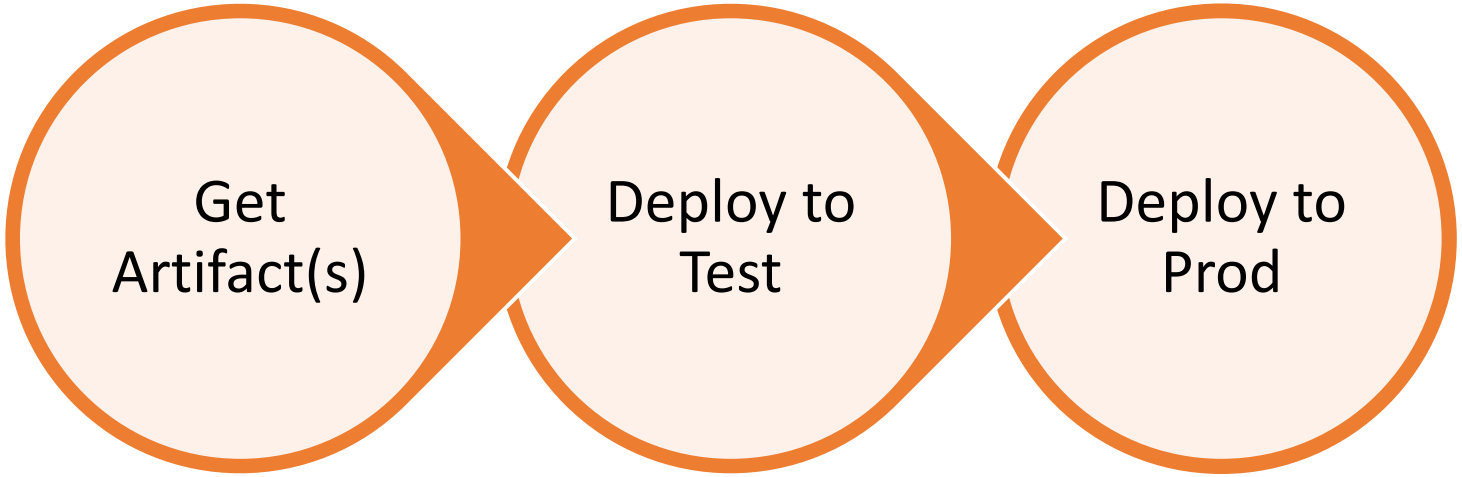
1. Pipeline – Export from Dev



2. Pipeline – Build Managed Solution



2. Pipeline – Release



Demo

ADO vs. GitHub

ADO vs. GitHub

- Source Control
- Run list of steps in predefined order
- orchestration engines

Features/Differences

	ADO	GitHub
Pipeline as code (YAML)	✓	✓
Pipeline UI	✓	✗
Omit unneeded structure	✓	✗
Multi stage in one file	✓	✗
Self hosted	✓ Agent	✓ Runner
“Fail fast” shell	✗	✓
PowerShell default	✗	✓
Deployment Groups	✓	✗
Variable Groups	✓	✗
Conditional Steps	“condition”	“if”
Condition Syntax	Functions (eq)	Infix notation (==)
Job dependencies	“dependsOn”	“needs”

YAML

- “Yet Another Markup Language”
- YAML is a human-readable data serialization language
- Industry standard
- Code
- Used by ADO & GitHub
- .yml or .yaml

Build Tools

- ADO: Power Platform Build Tools for Azure DevOps
  - https://learn.microsoft.com/en-us/power-platform/alm/devops-build-tools
  - Additional rights when executing environment management task
- GitHub: GitHub Actions for Microsoft Power Platform
  - https://learn.microsoft.com/en-us/power-platform/alm/devops-github-actions
- Others
  - Power Platform CLI – pac
    - https://learn.microsoft.com/en-us/power-platform/developer/cli/introduction
  - Power DevOps Tools
    - https://marketplace.visualstudio.com/items?itemName=WaelHamze.xrm-ci-framework-build-tasks

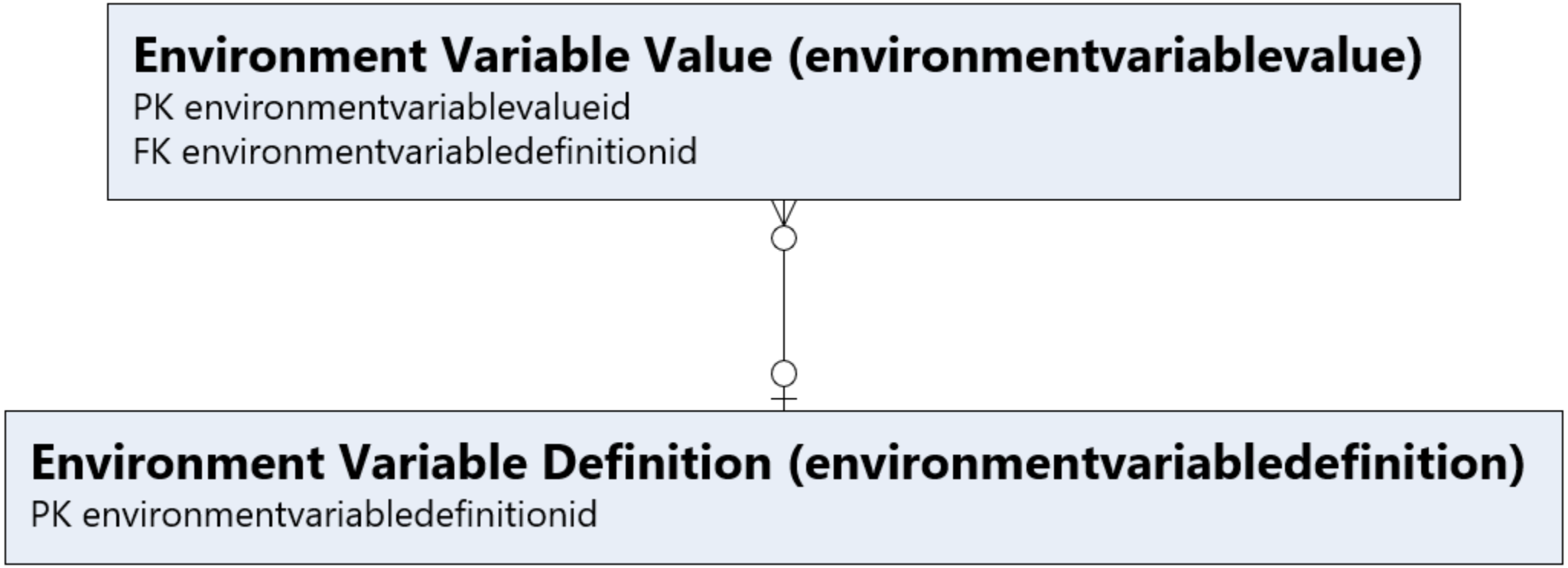
Questions?

Hands-on Lab #1

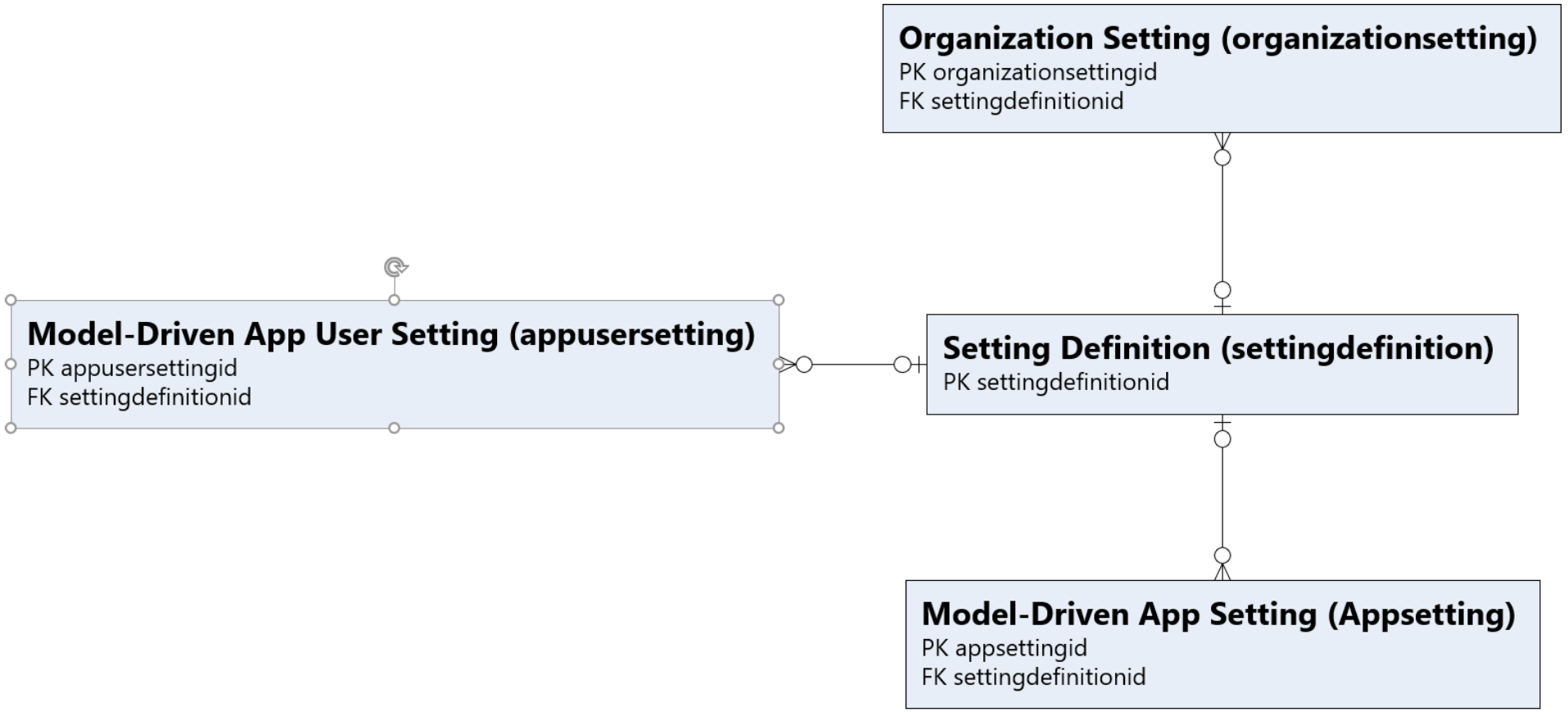
- Create Solution
- Create Basic ALM process

Environment Variables & Connection References

Environment Variables

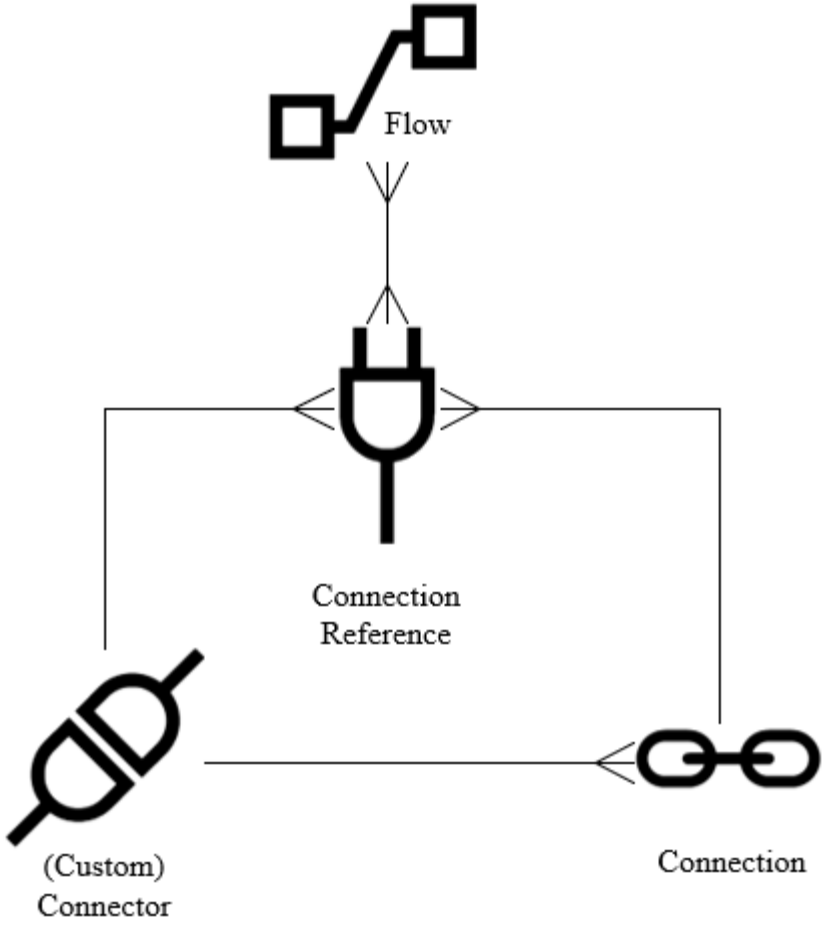


Settings


Organization Setting (organizationsetting)
PK organizationsettingid
FK settingdefinitionid
Model-Driven App User Setting (appusersetting)
PK appusersettingid
FK settingdefinitionid
Setting Definition (settingdefinition)
PK settingdefinitionid
Model-Driven App Setting (Appsetting)
PK appsettingid
FK settingdefinitionid

Connection References

- Connect connection with flow/app
- Solution aware



Settings file

- Automatically set EnvVar Values
- Automatically set ConRefs

- Share Canvas Apps

“Advanced” YAML

Advanced YAML

- Variables
- Parameters
- Templates
- Conditions
- Loops
- Expression Syntax

```yaml
- job: testCSharp
  displayName: Test CSharp
  condition: and(succeeded(), eq(lower(variables.containsCSharp), 'true'), eq(lower(variables.buildCode), 'true'))
```

```yaml
- ${{ each solution in split(variables.solutionNames, ',')}}:
  - template: Templates\ExportSolution.yml
    parameters:
      connection: $(connection)
      solutionName: ${{ solution }}
      version: $(Build.BuildNumber)
      handleCanvasApps: ${{ eq(lower(variables.handleCanvasApps), 'true') }}
      solutionType: $(UnpackSolutionType)
      localize: ${{ eq(lower(variables.localize), 'true') }}
```

Syntax	Example	When is it processed?	Where does it expand in a pipeline definition?	How does it render when not found?
macro	`$(var)`	runtime before a task executes	value (right side)	prints `$(var)`
template expression	`${{ variables.var }}`	compile time	key or value (left or right side)	empty string
runtime expression	`$[variables.var]`	runtime	value (right side)	empty string

Questions?

Hands-on Lab #2

- Add Settings
- Add Variables & Parameters

Versioning

Versioning

- Major.Minor.Build.Revision
- Many different approaches
  - Change last per build
  - Per Day
  - Manual
- Assemblies in sync
  - Problem first 2 elements

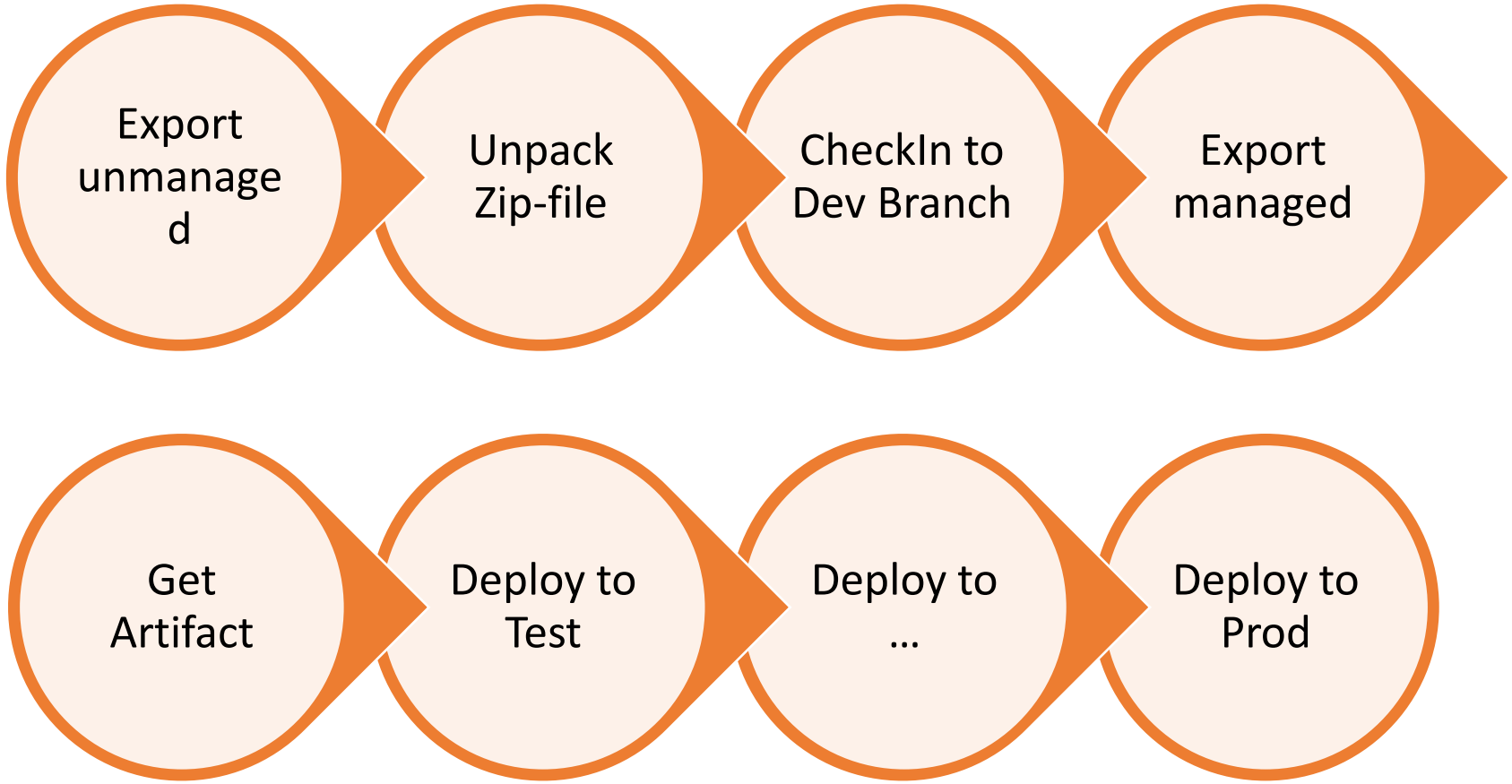
Quality Gates

Possibilities QGs

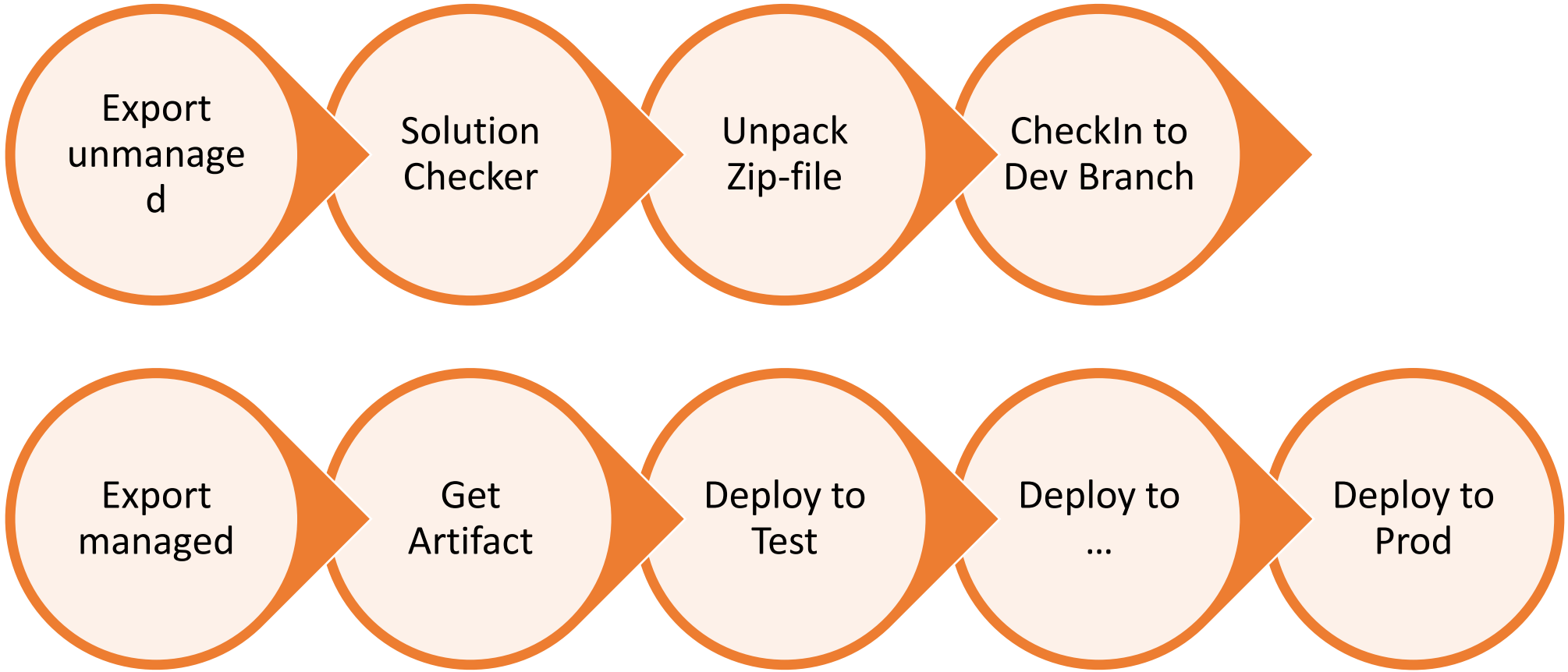
e.g.

- Solution Checker
- Unit Tests
- UI Tests (Playwright, EasyRepro)
- Integration Tests

Pipeline


Export unmanaged
Unpack Zip-file
CheckIn to Dev Branch
Export managed
Get Artifact
Deploy to Test
Deploy to …
Deploy to Prod

Pipeline – Adding QG



Run PAC in Pipeline

What is PAC

- Power Platform CLI
- Contains various operations for Power Platform
  - Environment lifecycle
  - Authentication
  - Solution packaging
  - Code components
  - …

Process of running in Pipeline

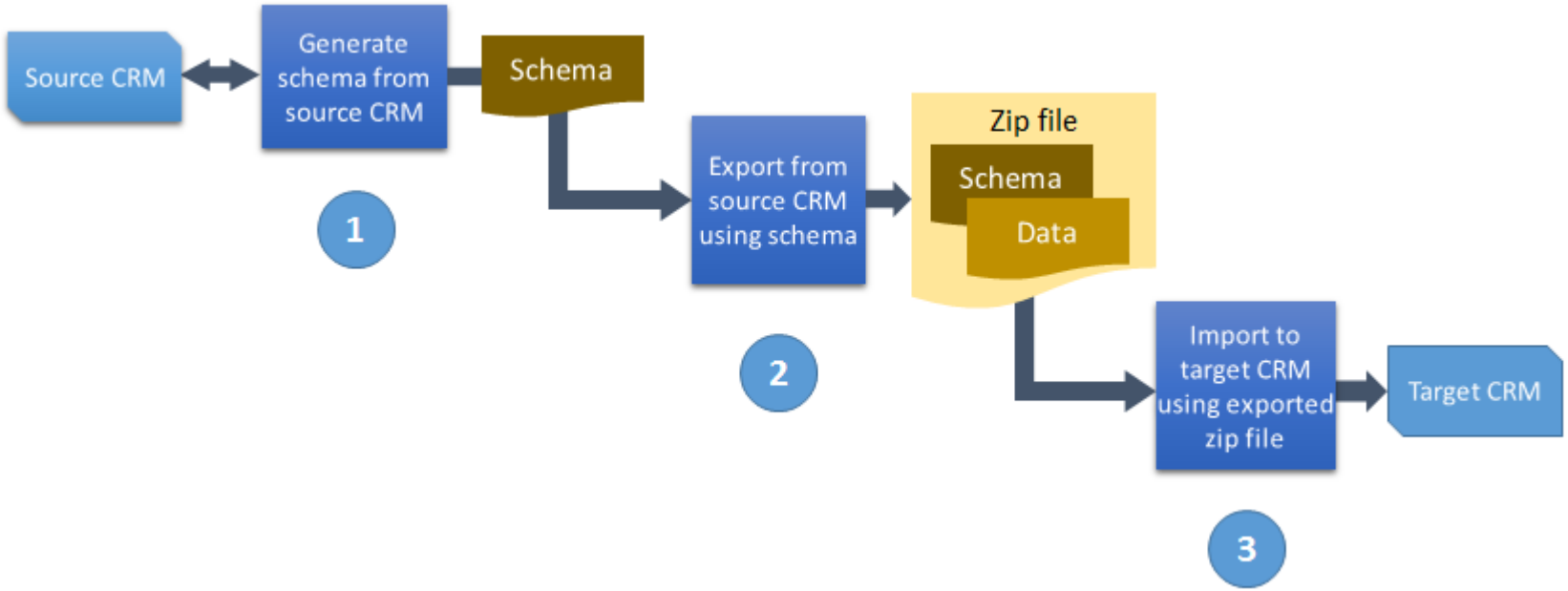
- Install NuGet
- Install NuGet PAC Package
- Run script to find pac
- Run pac

Move Data & Portal Configuration

Possibilities

- Data Migration Utility (DMU)
- Data Transporter in XTB
- Shuffle (in XTB)
- (PAC)

How DMU works



Process

- Create Schema file
- Generate ZIP with data in export pipeline
  - Export Dataverse Data Step
- Import data zip to target environment
  - Import Datverse Data Step

Project Setup

Folder structure

```
Development
├── spkl
├── (Azure)
├── Back-end
├── Front-end
PowerPlatform
├── CanvasApps
├── Config
│   ├── Schema
│   ├── Data
├── Solutions
PipelineDefinition
```

Projects

- Plugins
- Workflow Steps
- Applications

- TS
- PCF
- FE Test

Shared Projects

- Build in DLL
- Reusable code

- Plugin Base
- EarlyBound
- Shared Code

Pull Request

Steps

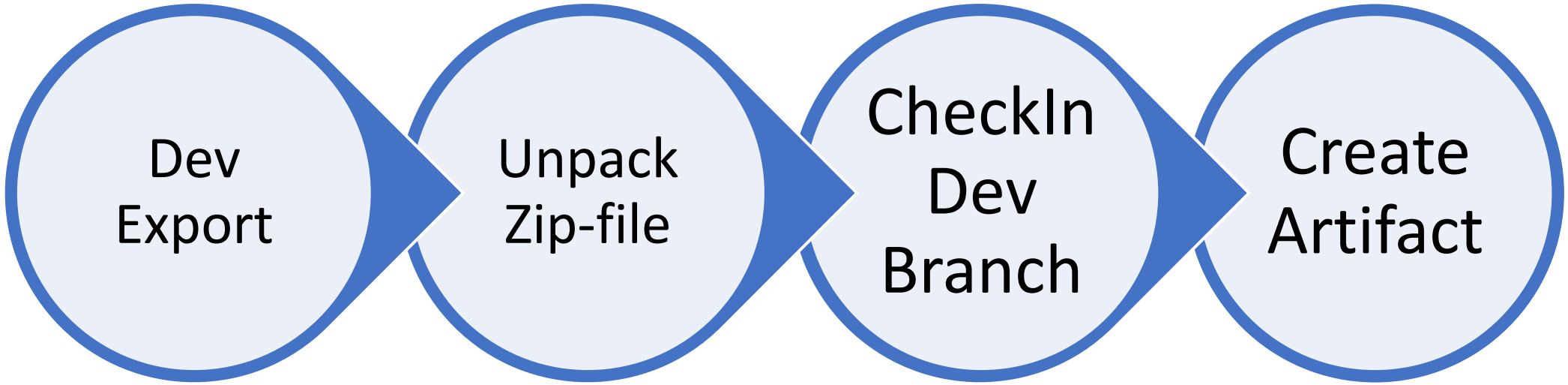
- Create new Pipeline
  - Install NuGet
  - Restore NuGet
  - Build
  - VSTest
- Add as policy in main branch

Other ALM approaches

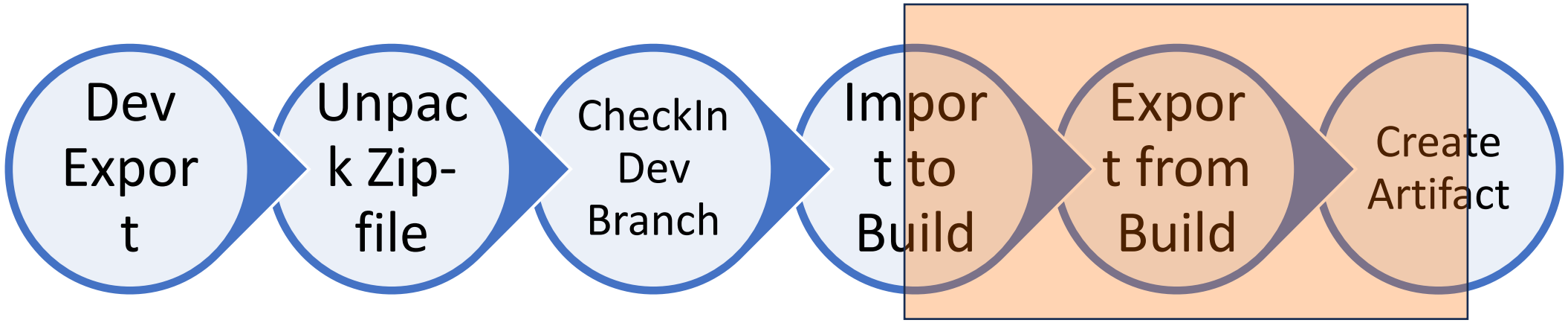
Other Approaches

- (JIT) Build Env
- Environment Centric
- Branches
- Different DEV environments

1. Pipeline – Create export from Dev



Export with (JIT) Build



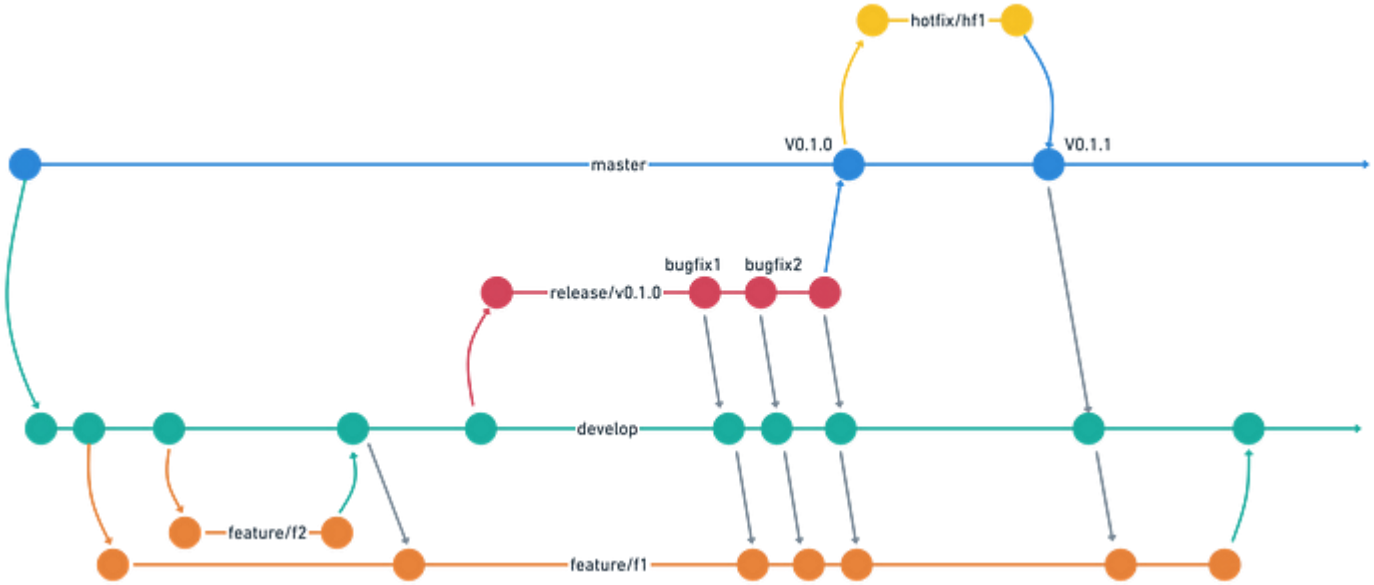
Pros & Cons of (JIT) Build

- Clean environment
- Detect dependencies
- Ensure clean solution

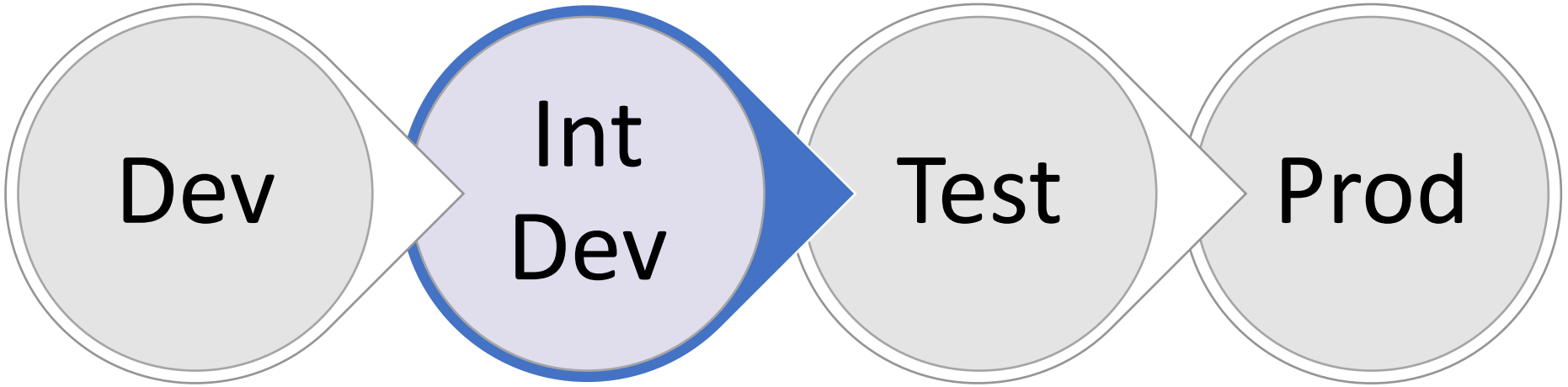
- Setup is slow
  - Additional languages
  - Additional Third-party solutions
- Unable to install first-party apps automatically

Branching

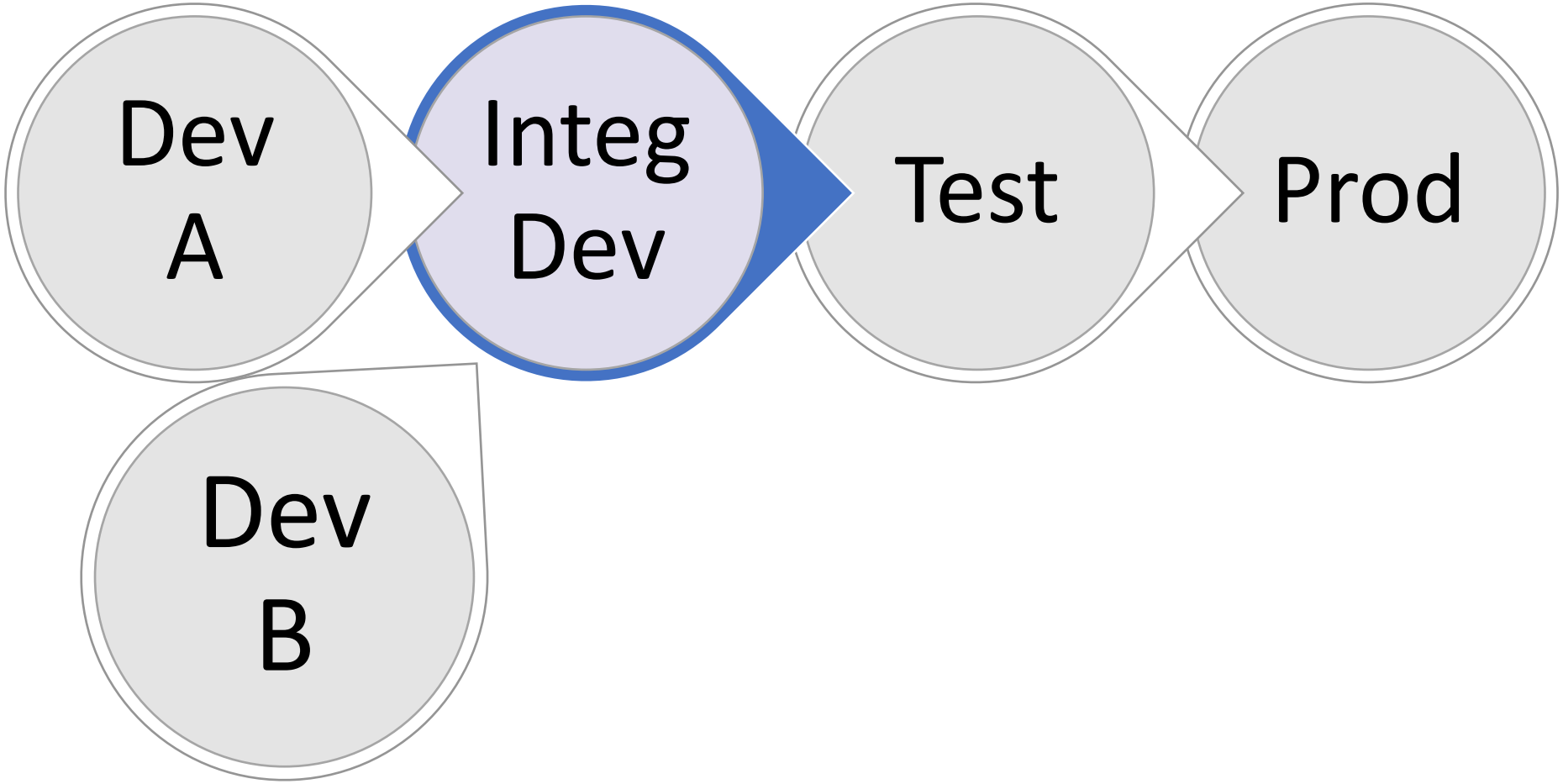
- Between one branch and branch per developer



Different DEV - Customer



Different DEV Teams



Ensure code state

Approaches

- Deploy latest to DEV
- Inject while packing (manually)
- Inject using mapping file

Process for deploying latest

- Build
- Run tests
- Deploy to DEV
  - Spkl

Process for manual injection

- Build
- Run tests
- Replace DLL and Webres in local repo
  - Deal with Ids
- Pack solution

Process for injection with mapping

- Unpack solution with mapping file
- Build
- Run tests
- Pack solution with mapping file

Package Deployer

Overview

- Tool to manage complex deployments
- Needed for AppSource
- Handles several Solutions
  - In order
- Can run custom C# script
- Able to inject Dll…

Process

- Create a Visual Studio or MSBuild project
- Add solutions and other files to the project
- Update provided HTML files (optional)
- Specify configuration values for the package
- Define custom code for the package
- Build and deploy the package

Thank you!

Twitter: https://twitter.com/BergmannBene

LinkedIn: https://www.linkedin.com/in/benedikt-bergmann

Mail: benedikt@benediktbergmann.eu

Blog: http://benediktbergmann.eu

European
Power Platform
Conference

cvent

PLEASE RATE THIS SESSION ON THE APP

GET IT ON Google Play

Download on the App Store