PowerShell + DevOps
Global Summit

PokaYoke

Mistake-Proofing your PowerShell scripts and functions

@Jaykul

April 7-10, 2025

Thank you Sponsors

Joel "Jaykul" Bennett

Principal DevOps Engineer

Solving problems with code
15x Microsoft MVP for PowerShell

github.com/Jaykul and PoshCode
discord.gg/PowerShell
HuddledMasses.org
@jaykul.powershell.social
@Jaykul@FOSStodon.org

What is Poka-Yoke?

What is Poka-Yoke?

ポカ	Poka	Mistake
ヨケ	Yoke	Avoidance

What is Poka-Yoke?

ポカ	Poka	Mistake
ヨケ	Yoke	Avoidance

A set of techniques and tools used to prevent errors or defects

Where do bugs come from?

Where do bugs come from?

Mistakes

Obviously. You didn't make them on purpose...

Mistakes Should Be

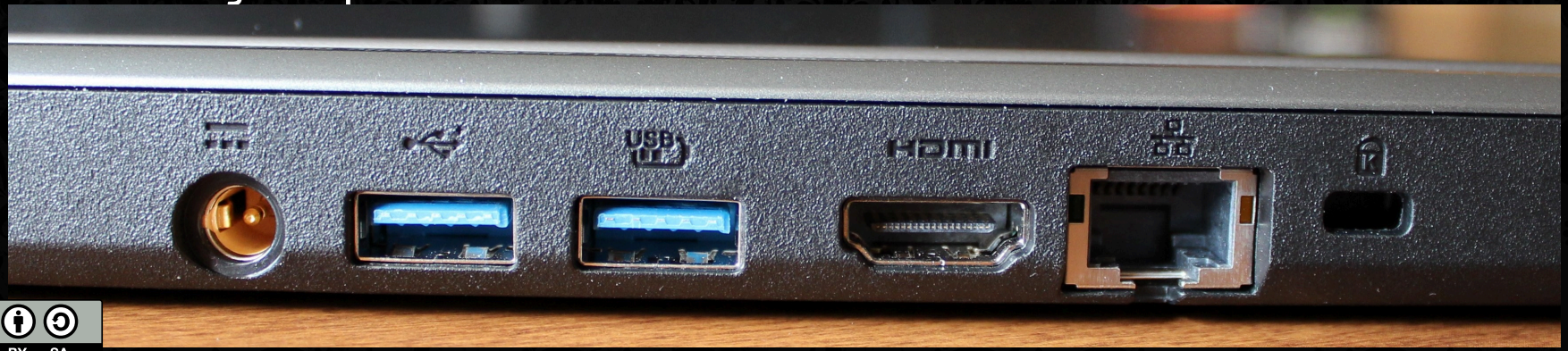
Mistakes Should Be

Obvious at a glance

Mistakes Should Be

Obvious at a glance

Basically impossible



Falling Into the Pit of Success

Consistency (noun): Reliability, uniformity, or conformity. Logical coherence. A singular way of doing things.

Falling Into the Pit of Success

Consistency (noun): Reliability, uniformity, or conformity. Logical coherence. A singular way of doing things.

- Approved Verbs

Falling Into the Pit of Success

Consistency (noun): Reliability, uniformity, or conformity. Logical coherence. A singular way of doing things.

- Approved Verbs
- Strongly Encouraged Guidelines

Falling Into the Pit of Success

Consistency (noun): Reliability, uniformity, or conformity. Logical coherence. A singular way of doing things.

- Approved Verbs
- Strongly Encouraged Guidelines
- Common Parameters

Falling Into the Pit of Success

Consistency (noun): Reliability, uniformity, or conformity. Logical coherence. A singular way of doing things.

- Approved Verbs
- Strongly Encouraged Guidelines
- Common Parameters
- Standard Parameter Names and Types

Falling Into the Pit of Success

Consistency (noun): Reliability, uniformity, or conformity. Logical coherence. A singular way of doing things.

- Approved Verbs
- Strongly Encouraged Guidelines
- Common Parameters
- Standard Parameter Names and Types
- Validating Parameter Input

Falling Into the Pit of Success

Consistency (noun): Reliability, uniformity, or conformity. Logical coherence. A singular way of doing things.

- Approved Verbs
- Strongly Encouraged Guidelines
- Common Parameters
- Standard Parameter Names and Types
- Validating Parameter Input
- Cmdlet Aliases

Falling Into the Pit of Success

Consistency (noun): Reliability, uniformity, or conformity. Logical coherence. A singular way of doing things.

- Approved Verbs
- Strongly Encouraged Guidelines
- Common Parameters
- Standard Parameter Names and Types
- Validating Parameter Input
- Cmdlet Aliases
- Parameter Aliases

Falling Into the Pit of Success

Consistency (noun): Reliability, uniformity, or conformity. Logical coherence. A singular way of doing things.

- Approved Verbs
- Strongly Encouraged Guidelines
- Common Parameters
- Standard Parameter Names and Types
- Validating Parameter Input
- Cmdlet Aliases
- Parameter Aliases
- Parameter Sets

Falling Into the Pit of Success

Consistency (noun): Reliability, uniformity, or conformity. Logical coherence. A singular way of doing things.

- Approved Verbs
- Strongly Encouraged Guidelines
- Common Parameters
- Standard Parameter Names and Types
- Validating Parameter Input
- Cmdlet Aliases
- Parameter Aliases
- Parameter Sets
- Supporting Wildcard Characters in Parameters

Falling Into the Pit of Success

Consistency (noun): Reliability, uniformity, or conformity. Logical coherence. A singular way of doing things.

- Approved Verbs
- Strongly Encouraged Guidelines
- Common Parameters
- Standard Parameter Names and Types
- Validating Parameter Input
- Cmdlet Aliases
- Parameter Aliases
- Parameter Sets
- Supporting Wildcard Characters in Parameters
- Input Filter Parameters

Falling Into the Pit of Success

Consistency (noun): Reliability, uniformity, or conformity. Logical coherence. A singular way of doing things.

- Approved Verbs
- Strongly Encouraged Guidelines
- Common Parameters
- Standard Parameter Names and Types
- Validating Parameter Input
- Cmdlet Aliases
- Parameter Aliases
- Parameter Sets
- Supporting Wildcard Characters in Parameters
- Input Filter Parameters
- Parameter Binding by Property Name

Falling Into the Pit of Success

Consistency (noun): Reliability, uniformity, or conformity. Logical coherence. A singular way of doing things.

- Approved Verbs
- Strongly Encouraged Guidelines
- Common Parameters
- Standard Parameter Names and Types
- Validating Parameter Input
- Cmdlet Aliases
- Parameter Aliases
- Parameter Sets
- Supporting Wildcard Characters in Parameters
- Input Filter Parameters
- Parameter Binding by Property Name
- About_Classes

Falling Into the Pit of Success

Consistency (noun): Reliability, uniformity, or conformity. Logical coherence. A singular way of doing things.

- Approved Verbs
- Strongly Encouraged Guidelines
- Common Parameters
- Standard Parameter Names and Types
- Validating Parameter Input
- Cmdlet Aliases
- Parameter Aliases
- Parameter Sets
- Supporting Wildcard Characters in Parameters
- Input Filter Parameters
- Parameter Binding by Property Name
- About_Classes
- Extending Output Objects

Falling Into the Pit of Success

Consistency (noun): Reliability, uniformity, or conformity. Logical coherence. A singular way of doing things.

- Approved Verbs
- Strongly Encouraged Guidelines
- Common Parameters
- Standard Parameter Names and Types
- Validating Parameter Input
- Cmdlet Aliases
- Parameter Aliases
- Parameter Sets
- Supporting Wildcard Characters in Parameters
- Input Filter Parameters
- Parameter Binding by Property Name
- About_Classes
- Extending Output Objects
- Custom Formatting Files

Make Mistakes Impossible

Make Mistakes Impossible

Use CmdletBinding to Support Common Parameters

```
class Player { [string]$Name; [Position]$Position; [string] ToString() { return $this.Name } }
class Position { [int]$X; [int]$Y; [string] ToString() { return "" + $this.X + ", " + $this.Y } }

function Move-Player {
    [CmdletBinding()]
    param(
    ...
```

Make Mistakes Impossible

Strongly Type Variables and Parameters

```
enum Direction { North; East; South; West }

function Move-Player {
    [CmdletBinding()]
    param(
        [Direction]$Direction,
        [int]$Distance,
        [switch]$Force
    )
    ...
```

Make Mistakes Impossible

Carefully Choose Parameter Names

```
enum Direction { North; East; South; West }

function Move-Player {
    [CmdletBinding()]
    param(
        [Direction]$CardinalDirection,
        [int]$DistanceInMeters,
        [switch]$Force
    )
    ...
```

Make Mistakes Impossible

Use Validation attributes

```
[ValidateSet("North", "East", "South", "West")]
[string]$CardinalDirection,

[ValidateScript({
    if ($CardinalDirection -eq "North" -and $Player.Position.Y - $_ -lt 0) {
        throw "You can't move that far North. The edge is only $($Position.Y) away."
    }
    $true
})]
[ValidateRange(1, 8)]
[int]$DistanceInMeters,

[ValidateScript({ Test-Path $_ })]
[string]$Path
```

Make Mistakes Impossible

Use ArgumentCompleter

```
[ValidateSet("North", "East", "South", "West")]
[Direction]$CardinalDirection,
[ArgumentCompleter({ param($Name, $Parameter, $Partial, $Ast, $Bound )
    @(switch($Bound.CardinalDirection) {
        North { ($Player.Position.Y)..0 }
        South { (8-$Player.Position.Y)..0 }
        East  { (8-$Player.Position.X)..0 }
        West  { ($Player.Position.X)..0 }
        default {
            ($Player.Position.X, $Player.Position.Y, (8 - $Player.Position.X), (8 - $Player.Position.Y) | Sort -Desc)[0]
        }
    }) | Where { $_ -gt 0 } # Zero isn't a valid move
})]
[int]$DistanceInMeters
...
```

See also … IValidateSetValuesGenerator, ArgumentCompletions

Make Mistakes Impossible

Support ShouldProcess for -WhatIf and -Confirm

```
[CmdletBinding(SupportsShouldProcess, ConfirmImpact="High")]
param(
    [Direction]$CardinalDirection,
    [int]$DistanceInMeters,
    [switch]$Force
)
switch($CardinalDirection) {
    North { $NewPosition = [Position]@{ X = $Player.Position.X; Y = $Player.Position.Y - $DistanceInMeters; } }
    East  { $NewPosition = [Position]@{ X = $Player.Position.X + $DistanceInMeters; Y = $Player.Position.Y } }
    South { $NewPosition = [Position]@{ X = $Player.Position.X; Y = $Player.Position.Y + $DistanceInMeters;  } }
    West  { $NewPosition = [Position]@{ X = $Player.Position.X - $DistanceInMeters; Y = $Player.Position.Y } }
}
if ($Force -or $PSCmdlet.ShouldProcess($Player.Name, "Move ${DistanceInMeters} meters ${CardinalDirection} to ${NewPosi
    ...
```

Never Repeat a Mistake

Never Repeat a Mistake

Write Regression Tests

Never Repeat a Mistake

Write Regression Tests

- Made a mistake?

Never Repeat a Mistake

Write Regression Tests

- Made a mistake?
- Write a test!

Never Repeat a Mistake

Write Regression Tests

- Made a mistake?
- Write a test!
- Fix the mistake

Never Repeat a Mistake

Write Regression Tests

- Made a mistake?
- Write a test!
- Fix the mistake
- Integrate the test

Never Repeat a Mistake

Write Regression Tests

- Made a mistake?
- Write a test!
- Fix the mistake
- Integrate the test

Use PSScriptAnalyzer

Never Repeat a Mistake

Write Regression Tests

- Made a mistake?
- Write a test!
- Fix the mistake
- Integrate the test

Use PSScriptAnalyzer

Consistency (noun): Reliability, uniformity, or conformity. Logical coherence. A singular way of doing things.

Never Repeat a Mistake

Write Regression Tests

- Made a mistake?
- Write a test!
- Fix the mistake
- Integrate the test

Use PSScriptAnalyzer

Consistency (noun): Reliability, uniformity, or conformity. Logical coherence. A singular way of doing things.

Consistency facilitates intuition

Never Repeat a Mistake

Write Regression Tests

- Made a mistake?
- Write a test!
- Fix the mistake
- Integrate the test

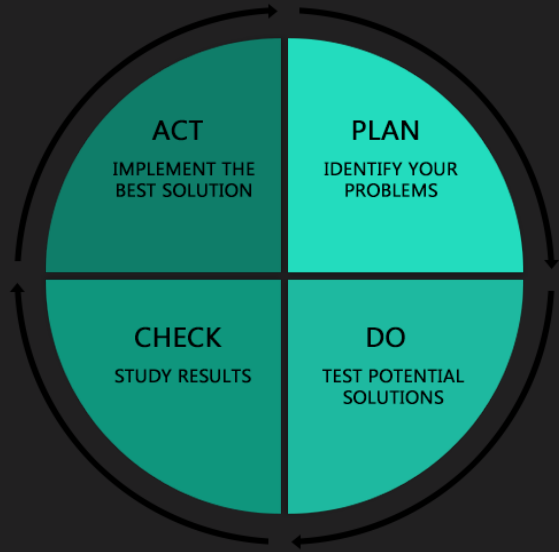
Use PSScriptAnalyzer

Consistency (noun): Reliability, uniformity, or conformity. Logical coherence. A singular way of doing things.

Consistency facilitates intuition

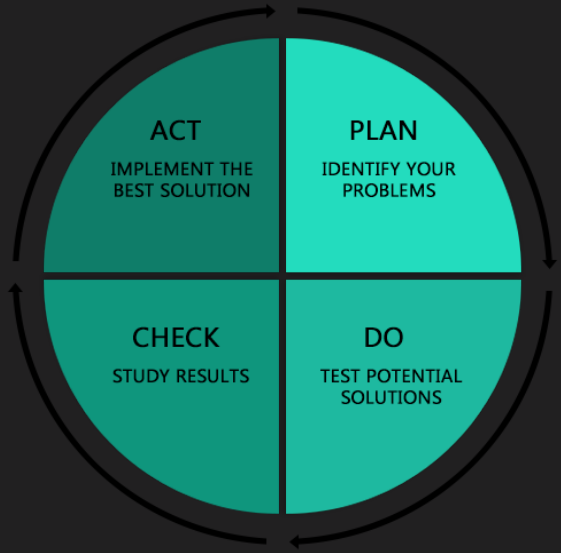
Consistency increases maintainability

Continuous Improvement



Continuous Improvement

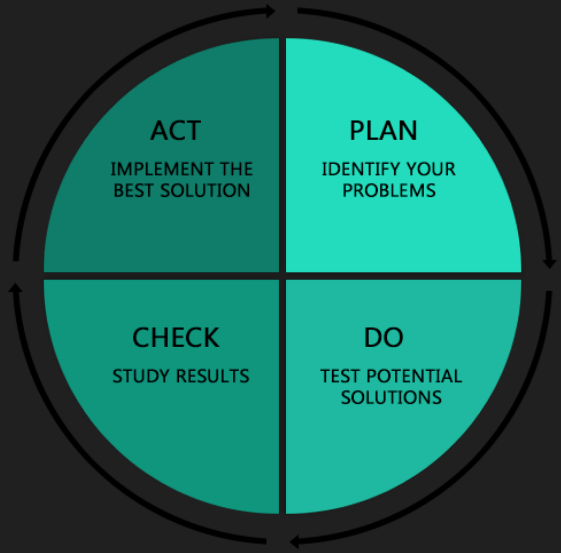
Plan-Do-Check-Act



Continuous Improvement

Plan-Do-Check-Act

1. **Plan**: Identify your (possible) problems

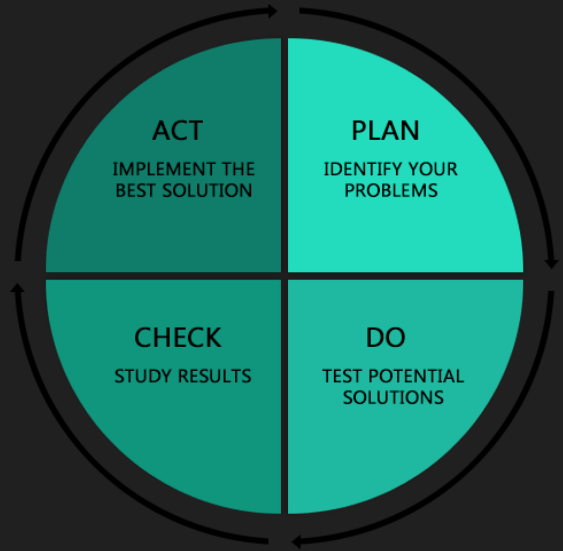


Continuous Improvement

Plan-Do-Check-Act

1. **Plan**: Identify your (possible) problems

    The Five Whys

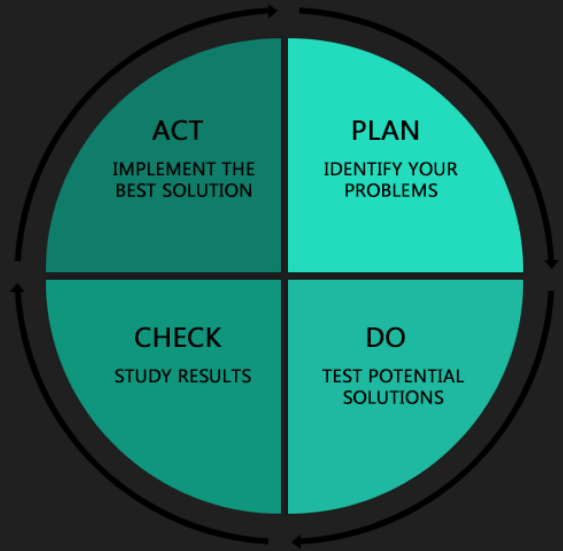


Continuous Improvement

Plan-Do-Check-Act

1. **Plan**: Identify your (possible) problems

    The Five Whys Ask Why, Five Times.

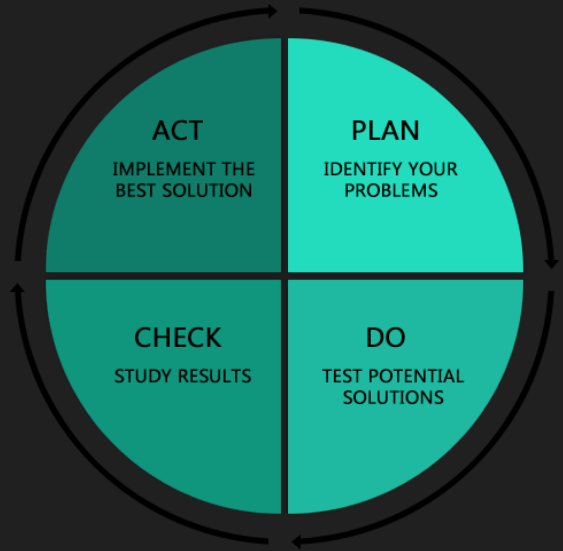


Continuous Improvement

Plan-Do-Check-Act

1. **Plan**: Identify your (possible) problems

   The Five Whys Ask Why, Five Times.
   Our deployments are failing



Continuous Improvement

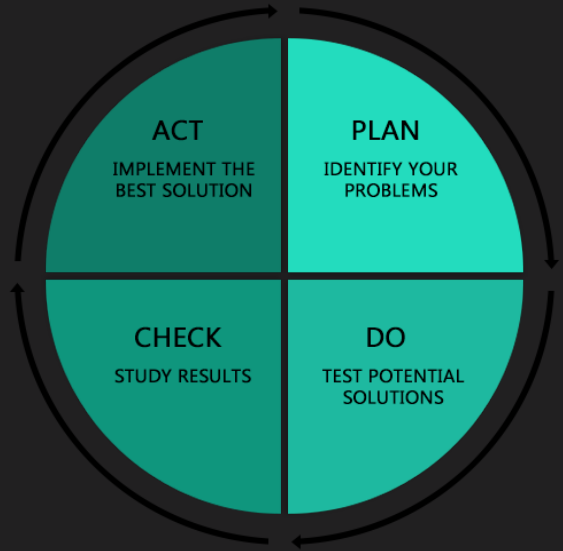
Plan-Do-Check-Act

1. **Plan**: Identify your (possible) problems

   The Five Whys Ask Why, Five Times.

   Our deployments are failing

   1. Why? Our "publish" step is failing to upload.



Continuous Improvement

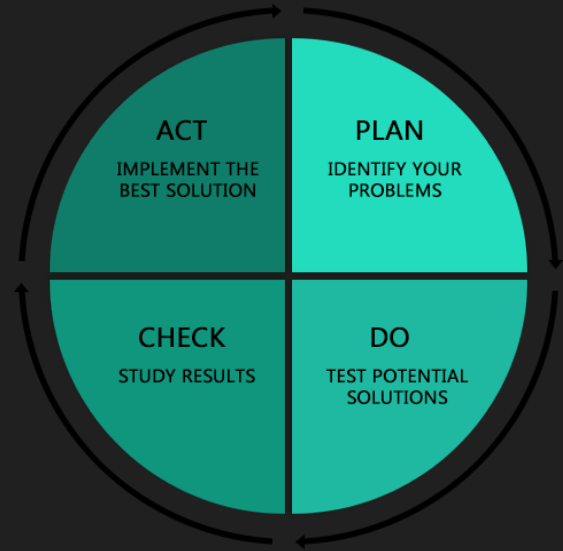
Plan-Do-Check-Act

1. **Plan**: Identify your (possible) problems

   The Five Whys Ask Why, Five Times.

   Our deployments are failing

   1. Why? Our "publish" step is failing to upload.
   2. Why? Some sort of error authenticating to the service.



Continuous Improvement

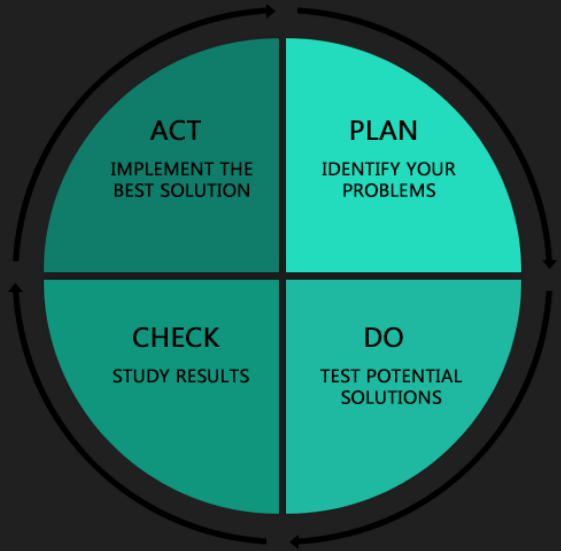
Plan-Do-Check-Act

1. **Plan**: Identify your (possible) problems

The Five Whys Ask Why, Five Times.

Our deployments are failing

1. Why? Our "publish" step is failing to upload.
2. Why? Some sort of error authenticating to the service.
3. Why? It turns out the credentials are wrong.



Continuous Improvement

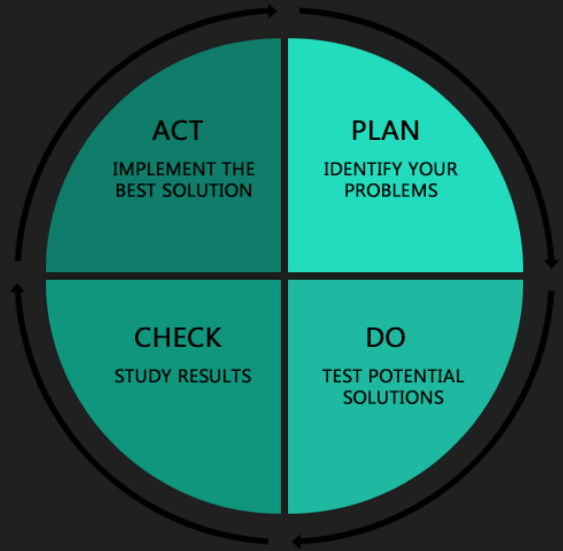
Plan-Do-Check-Act

1. **Plan**: Identify your (possible) problems

The Five Whys Ask Why, Five Times.

Our deployments are failing

1. Why? Our "publish" step is failing to upload.
2. Why? Some sort of error authenticating to the service.
3. Why? It turns out the credentials are wrong.
4. Why? Well, the password expired.



Continuous Improvement

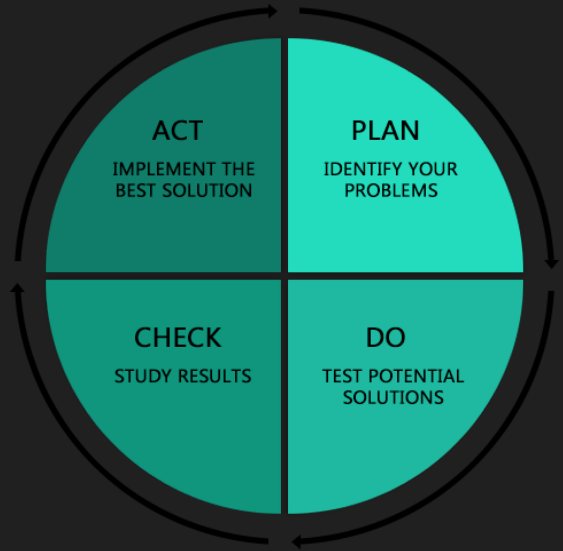
Plan-Do-Check-Act

1. **Plan**: Identify your (possible) problems

   The Five Whys Ask Why, Five Times.

   Our deployments are failing

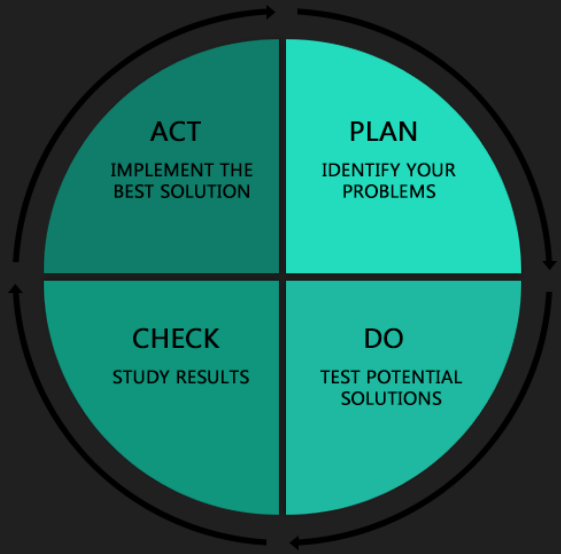
   1. Why? Our "publish" step is failing to upload.
   2. Why? Some sort of error authenticating to the service.
   3. Why? It turns out the credentials are wrong.
   4. Why? Well, the password expired.
   5. Why? We forgot to change the password ahead of time.



Continuous Improvement

Plan-Do-Check-Act

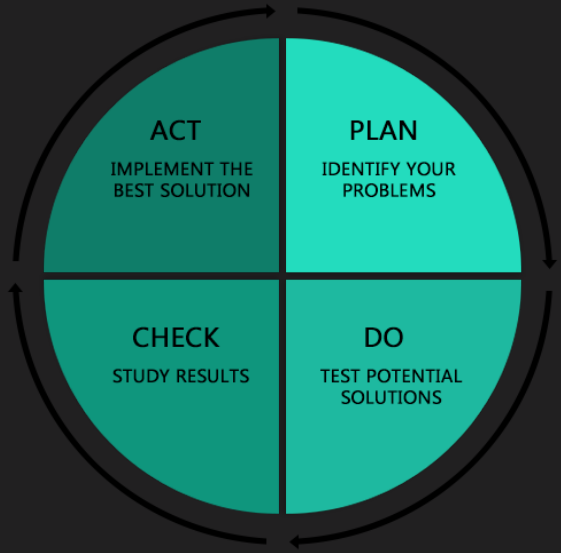
1. **Plan**: Identify your problems
   … and possible solutions
   … and how to measure your results.



Continuous Improvement

Plan-Do-Check-Act

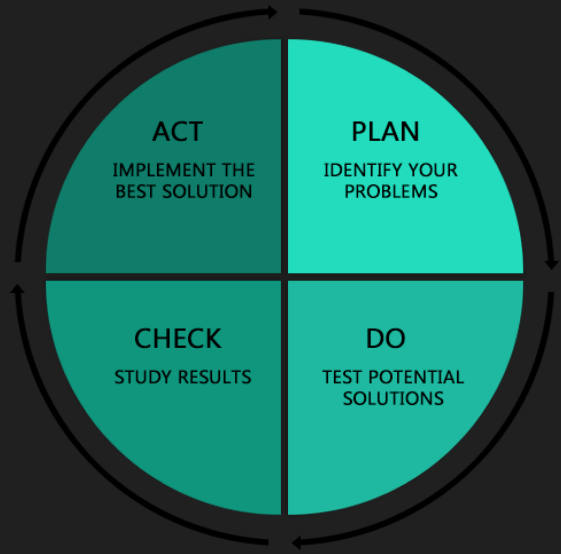
1. **Plan**: Identify your problems
   … and possible solutions
   … and how to measure your results.
2. **Do**: Implement and test (mistake-proofing) solutions.



Continuous Improvement

Plan-Do-Check-Act

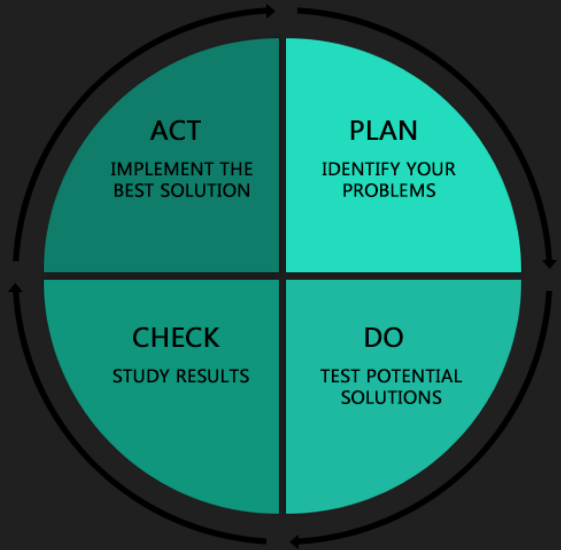
1. **Plan**: Identify your problems
   … and possible solutions
   … and how to measure your results.
2. **Do**: Implement and test (mistake-proofing) solutions.
3. **Check**: Study the results. Compare solutions.



Continuous Improvement

Plan-Do-Check-Act

1. **Plan**: Identify your problems
   … and possible solutions
   … and how to measure your results.
2. **Do**: Implement and test (mistake-proofing) solutions.
3. **Check**: Study the results. Compare solutions.
4. **Act**: Adjust. Implement the best solution. Document.
   Train your team(s).



Design By Writing ... Help F1

Plan:

Do:

Check:

Act:

Design By Writing ... Help F1

Plan: Write .EXAMPLEs for Common Scenarios

Do:

Check:

Act:

Design By Writing ... Help F1

Plan: Write .EXAMPLEs for Common Scenarios

Do: Implement Your Examples

Check:

Act:

Design By Writing ... Help F1

Plan: Write .EXAMPLEs for Common Scenarios

Do: Implement Your Examples

Check: Test Your Examples

Act:

Design By Writing ... Help F1

Plan: Write .EXAMPLEs for Common Scenarios

Do: Implement Your Examples

Check: Test Your Examples

Act: Document Your Interface ALT + H

Detecting Mistakes

```powershell
function Move-Player {
    [CmdletBinding()]
    param()
    try {
        $MyInvocation.Instrument()
        Your-Code-Goes-Here
    # Special handling for errors you expect
    } catch [System.IO.FileNotFoundException] {
        Write-Error "The target game file was not found." -Recommend "Check the path and try again."
    } catch {
        Write-ErrorInformation -InformationAction $InformationPreference -InformationVariable global:MyGame_ErrorInforma
        Write-Error $_ -Recommend ("Please consider reporting this issue at https://github.com/Jaykul/DevOps2025/issues`
                "  Include as much information as you're comfortable sharing from `$MyGame_ErrorInformation | Set-Clipbo
    } finally {
        $MyInvocation.TraceEnd()
    }
}
```

Detecting Mistakes

Error Handling & Logging

```powershell
function Move-Player {
    [CmdletBinding()]
    param()
    try {
        $MyInvocation.Instrument()
        Your-Code-Goes-Here
    # Special handling for errors you expect
    } catch [System.IO.FileNotFoundException] {
        Write-Error "The target game file was not found." -Recommend "Check the path and try again."
    } catch {
        Write-ErrorInformation -InformationAction $InformationPreference -InformationVariable global:MyGame_ErrorInforma
        Write-Error $_ -Recommend ("Please consider reporting this issue at https://github.com/Jaykul/DevOps2025/issues`
                "  Include as much information as you're comfortable sharing from `$MyGame_ErrorInformation | Set-Clipbo
    } finally {
        $MyInvocation.TraceEnd()
    }
}
```

Detecting Mistakes

Error Handling & Logging

```powershell
function Move-Player {
    [CmdletBinding()]
    param()
    try {
        $MyInvocation.Instrument()
        Your-Code-Goes-Here
    # Special handling for errors you expect
    } catch [System.IO.FileNotFoundException] {
        Write-Error "The target game file was not found." -Recommend "Check the path and try again."
    } catch {
        Write-ErrorInformation -InformationAction $InformationPreference -InformationVariable global:MyGame_ErrorInforma
        Write-Error $_ -Recommend ("Please consider reporting this issue at https://github.com/Jaykul/DevOps2025/issues`
                "  Include as much information as you're comfortable sharing from `$MyGame_ErrorInformation | Set-Clipbo
    } finally {
        $MyInvocation.TraceEnd()
    }
}
```

Detecting Mistakes

Error Handling & Logging

```powershell
function Move-Player {
    [CmdletBinding()]
    param()
    try {
        $MyInvocation.Instrument()
        Your-Code-Goes-Here
    # Special handling for errors you expect
    } catch [System.IO.FileNotFoundException] {
        Write-Error "The target game file was not found." -Recommend "Check the path and try again."
    } catch {
        Write-ErrorInformation -InformationAction $InformationPreference -InformationVariable global:MyGame_ErrorInforma
        Write-Error $_ -Recommend ("Please consider reporting this issue at https://github.com/Jaykul/DevOps2025/issues`
                "  Include as much information as you're comfortable sharing from `$MyGame_ErrorInformation | Set-Clipbo
    } finally {
        $MyInvocation.TraceEnd()
    }
}
```

THANK YOU!

Feedback is a gift. Please review this session!