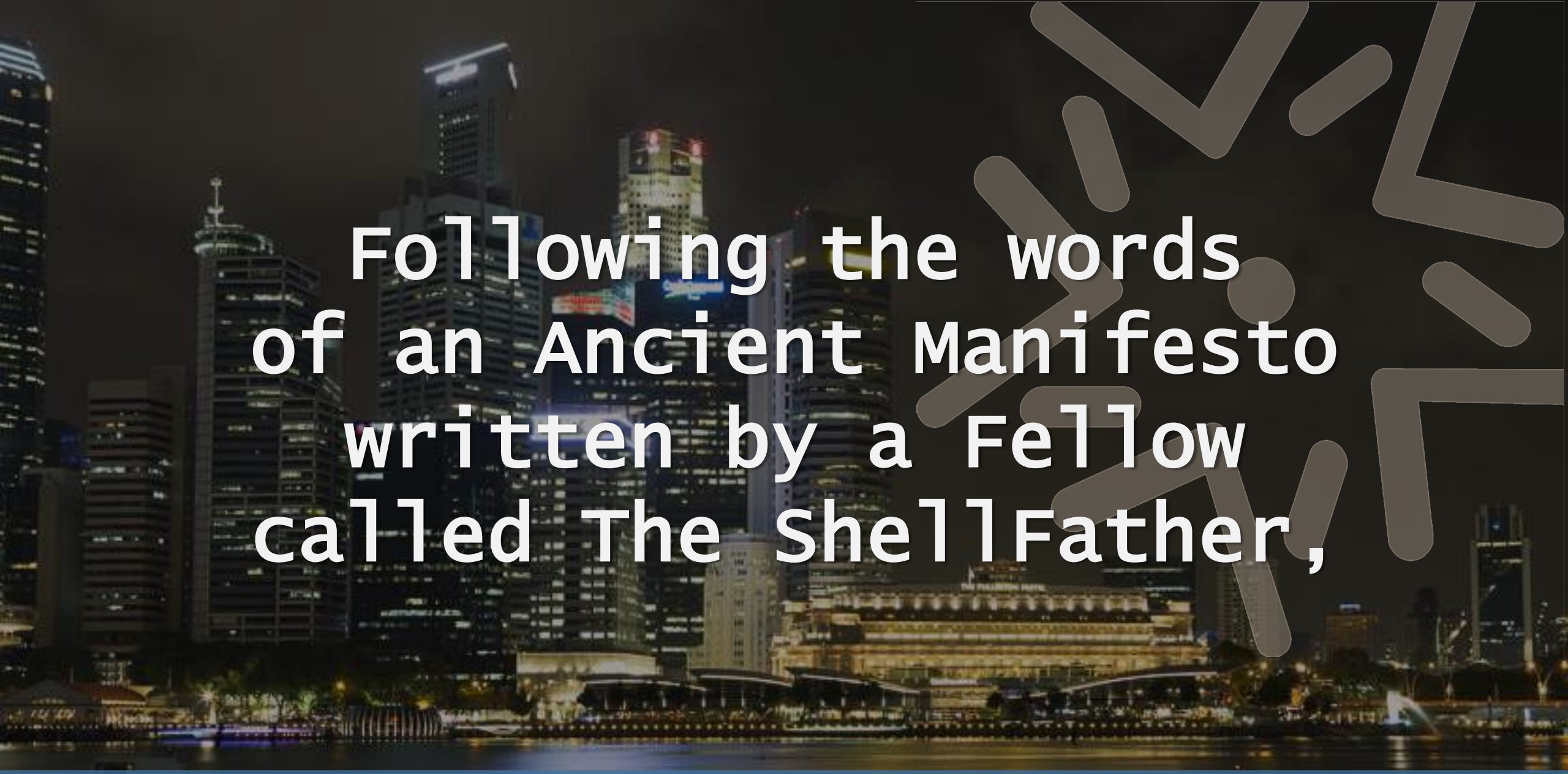
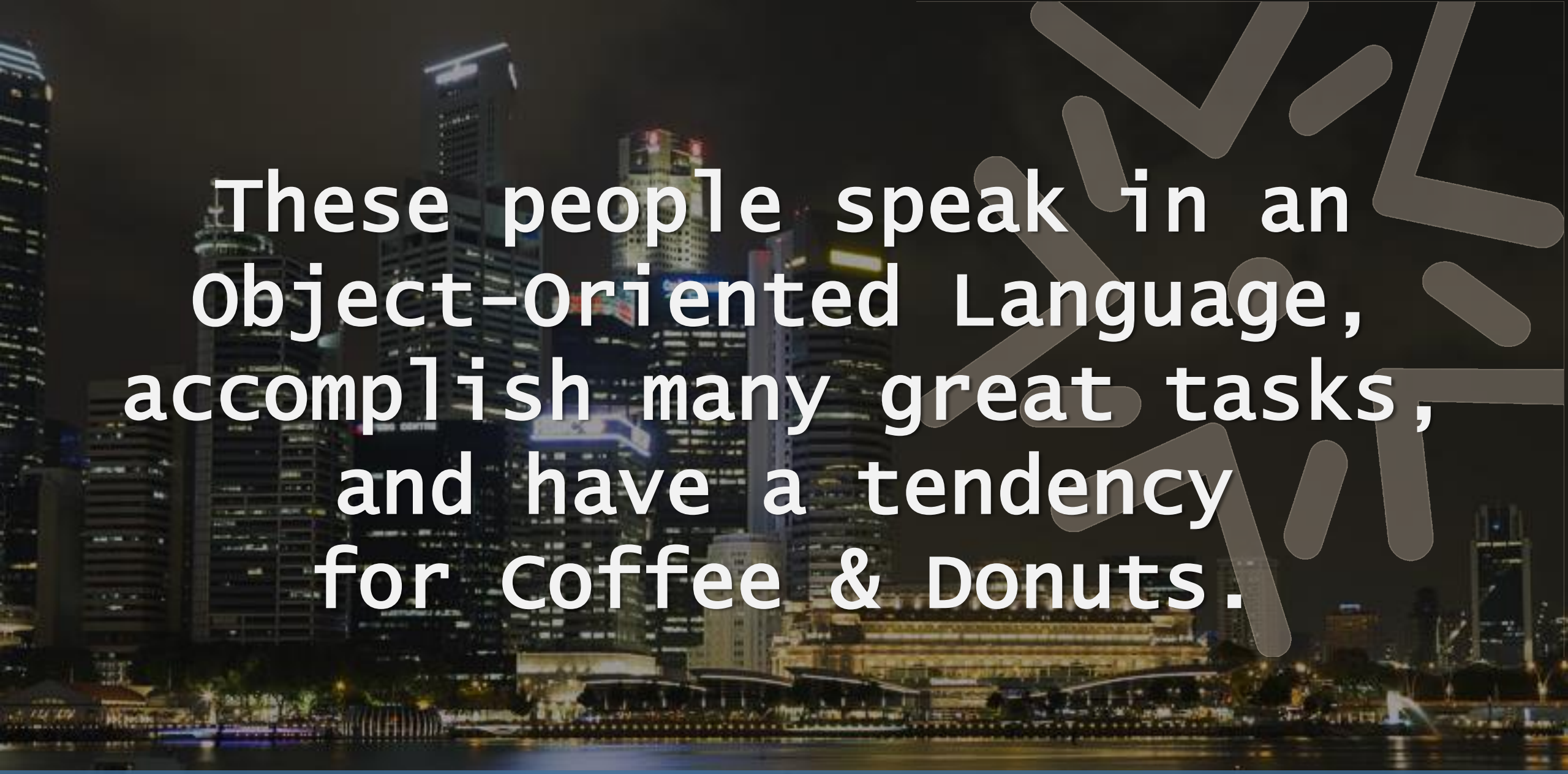
PowerShell Conference
Singapore 2017

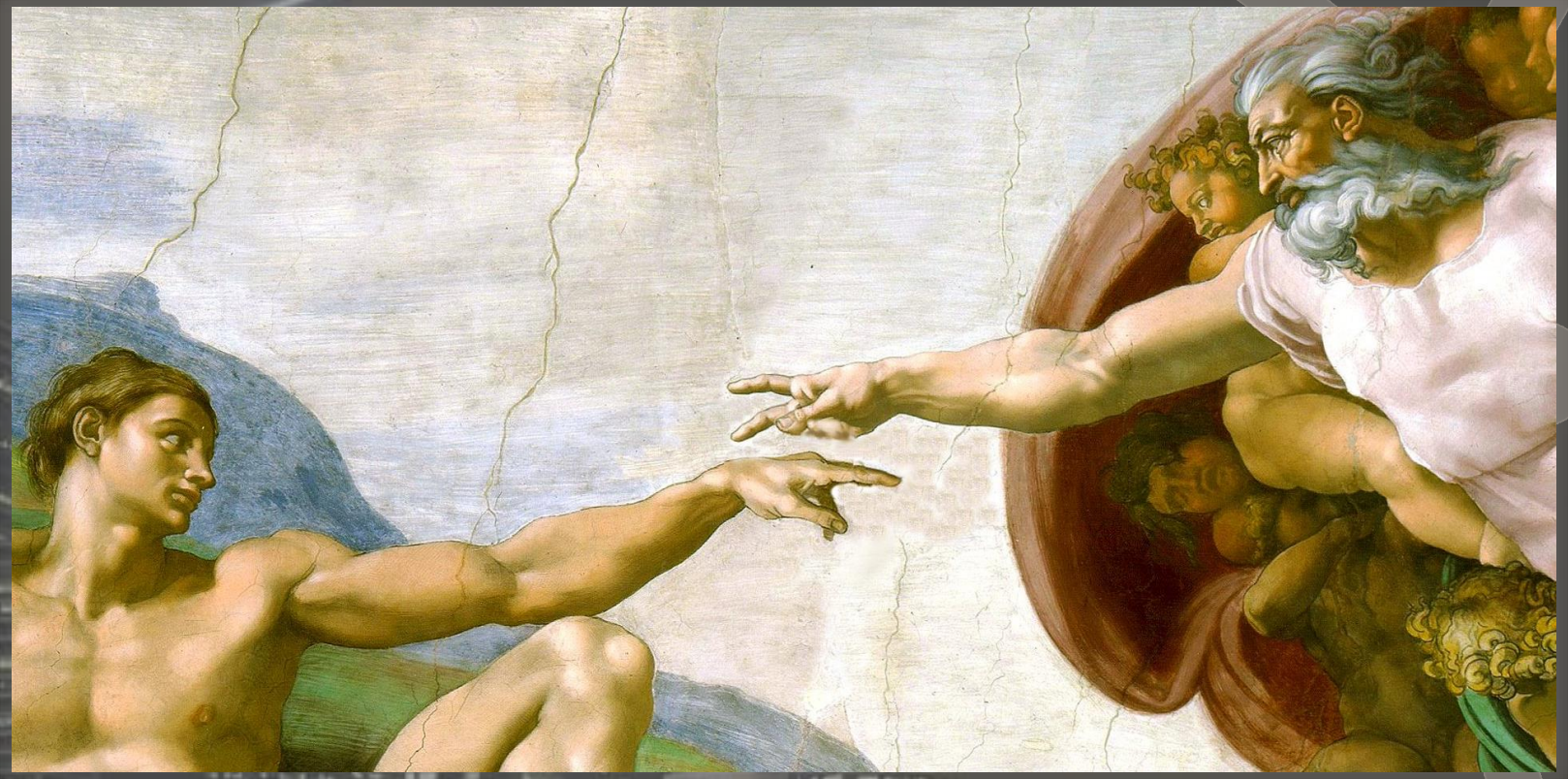

PowerShell Conference
Singapore 2017
Yo Bro!
Do You Even Parameterize ?
Microsoft
KEMP
APPLICATION DELIVERY
SAPIEN
Technologies, Inc.
CHEF
CHEF.IO
Konfx.
www.konfx.io


INTRO...

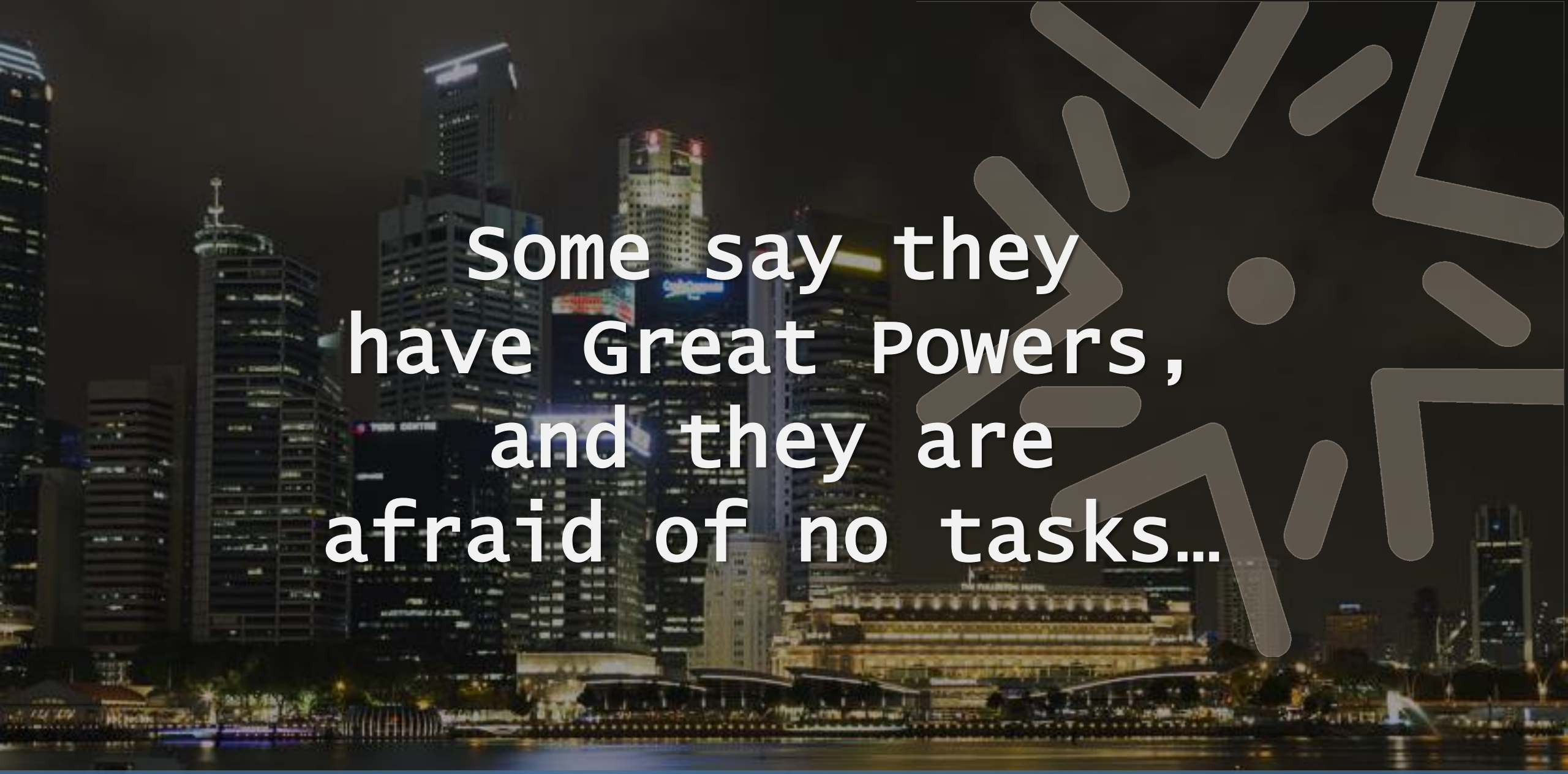

Somewhere in our
Digital Society,
is a Community
Of Powerful Builders.


Following the words
of an Ancient Manifesto
written by a Fellow
called The ShellFather,


These people speak in an
Object-Oriented Language,
accomplish many great tasks,
and have a tendency
for Coffee & Donuts.



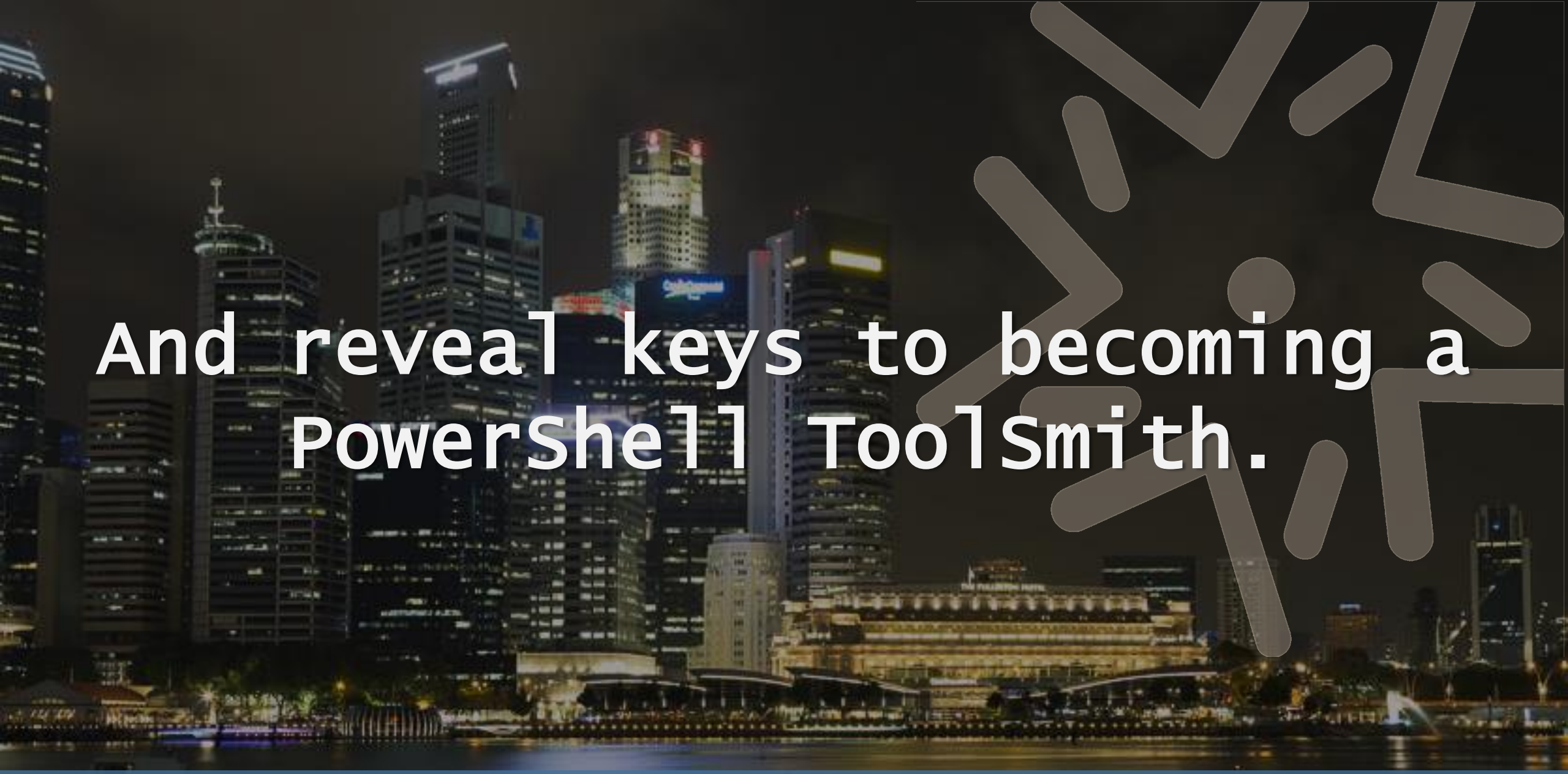
They even have a Secret Handshake.


Some say they
have Great Powers,
and they are
afraid of no tasks…


Other say the ShellFather
wears Toe Socks,
and has more than
1337 Ties.

Nobody knows…


But in this Presentation,
We will share some of their
secret tradecraft,


And reveal keys to becoming a
PowerShell ToolSmith.

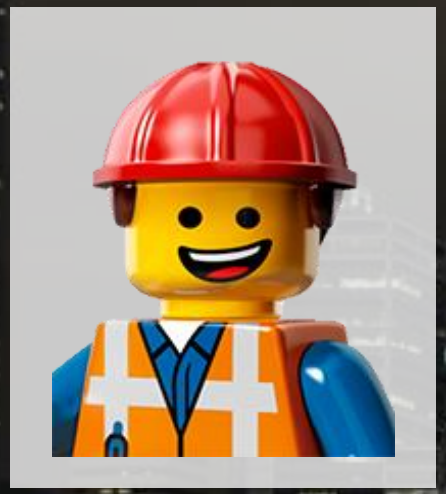
Yo Bro!
Do You Even Parameterize ?

about_Building_PowerShell_Tools

@SadProcessor

PowerShell Conference
Singapore 2017

WHOIS



Walter Legowski

@SadProcessor

PowerShell Automation Engineer

- Lego
- Tools
- PowerShell

DISCLAIMER

Been a n00b since IT.
(Decided not to change)

Can’t Type nor Click.
+ Strange Coding Style (Ratliff)

Make mistakes as much as possible.
(Will make some during Demos)

+ Tool presented is still Beta.

TOPIC OF THIS TALK

TOOLS...

TOPIC OF THIS TALK

USERS...

TOPIC OF THIS TALK

MAKERS...

TOPIC OF THIS TALK

- User Input
- User friendliness
- Dev friendliness

Write Less Code & Get More…

GOAL OF THIS TALK

- Review Framework Features
- Share Personal Workflow
- Demo Homemade Tools

Show why PowerShell is Awesome for Tool-Makers

1 - RTFM EXPRESS

2 - SCRIPTING ON SCRIPTONITE

3 - SECRETS OF THE DONUT MASTER

1 - RTFM EXPRESS

```
# Basic Recipe
# Cmdlet Bindings
# Parameter Validation
# Value From Pipeline
# Parameter Sets
# Dynamic Parameters
```

Function ??

TL;DR:

A sequence of PowerShell statements you like so much you give it a Name, and when you call that Name the Commands run in Sequence.

[help about_Functions]

Function

```
# Take Your Favorite Code
<# Code Goes Here #>
```

Function

```
# Make it a Function
function DoStuff{
    <# Code Goes Here #>
    }
```


Done.

Function

```
# Call it > Code runs
function DoStuff{
    <# Code Goes Here #>
    }
```

```
PS> DoStuff
```

Function

```
# Pass Arguments
function DoStuff{
    <# Code Goes Here #>
    }
```

```
PS> DoStuff  123
```

Function

```
# Pass Arguments
function DoStuff{
    <# Code Goes Here #>
    }
```

```
PS> DoStuff  abc
```

Function

```
# Pass Arguments
function DoStuff{
    <# Code Goes Here #>
    }
```

```
PS> DoStuff  abc 123
```

Function

```
# Argument Values stored in $Args
function DoStuff{
    <# Code Goes Here #>
    }
```

```
PS> DoStuff  abc 123 ...
```

$Args

Function

```
# Argument Values stored in $Args
function DoStuff{
    <# Code Goes Here #>
    }
```

```
PS> DoStuff  abc   123   ...
```

$Args[0] $Args[1] $Args[2]


Not Great ...


We can do
waaaaaay better than this ...

Matt Graeber
@mattifestation

Following

#PowerShell function parameter design fulfills nearly all of my OCD tendencies.

5:20 PM - 20 May 2017

2 Retweets **9** Likes

2 2 9

PoSh

```
Function DoStuff{
    <# Code Goes Here #>
    }
```

PoSh

Verb-Noun Syntax

```
Function Invoke-DoStuff{
    <# Code Goes Here #>
    }
```

PoSh

Named Parameters

```
Function Invoke-DoStuff{
    Param(
        $ParamOne,
        $Paramtwo
        )
    <# Code Goes Here #>
    }
```

PoSh Parameter Description

```
Function Invoke-DoStuff{
    Param(
        # This is Param One Description
        $ParamOne,
        # This is Param Two Description
        $Paramtwo
        )
    <# Code Goes Here #>
    }
```

PoSh

Help

```
<#
.Synopsis
   Do Some Stuff
.DESCRIPTION
   Long Description
.EXAMPLE
   Invoke-DoStuff
   Example Description
#>
Function Invoke-DoStuff{
    Param(
       # This is Param One Description
       $ParamOne,
       # This is Param Two Description
       $Paramtwo
```

PoSh Help

```
<#...#>
Function Invoke-DoStuff{
    Param(
       # This is Param One Description
       $ParamOne,
       # This is Param Two Description
       $Paramtwo
       )
    <# Code Goes Here #>
    }
```

PoSh

Function Alias

```
<#...#>
Function Invoke-DoStuff{
    [Alias('DoStuff')]
    Param(
       # This is Param One Description
       $ParamOne,
       # This is Param Two Description
       $Paramtwo
       )
    <# Code Goes Here #>
    }
```

PoSh Parameter Type

```
<#...#>
Function Invoke-DoStuff{
    [Alias('DoStuff')]
    Param(
       # This is Param One Description
       [string]$ParamOne,
       # This is Param Two Description
       [int]$Paramtwo
       )
    <# Code Goes Here #>
    }
```

PoSh Parameter Alias

```
<#...#>
Function Invoke-DoStuff{
    [Alias('DoStuff')]
    Param(
       # This is Param One Description
       [Alias('P1')][string]$ParamOne,
       # This is Param Two Description
       [Alias('P2')][int]$Paramtwo
       )
    <# Code Goes Here #>
    }
```

PoSh Parameter Attributes

```
<#...#>
Function Invoke-DoStuff{
    [Alias('DoStuff')]
    Param(
       # This is Param One Description
       [Parameter()][Alias('P1')][string]$ParamOne,
       # This is Param Two Description
       [Parameter()][Alias('P2')][int]$Paramtwo
       )
    <# Code Goes Here #>
    }
```

PoSh Parameter Attributes

```
<#...#>
Function Invoke-DoStuff{
    [Alias('DoStuff')]
    Param(
       # This is Param One Description
       [Parameter()][Alias('P1')][string]$ParamOne,
       # This is Param Two Description
       [Parameter()][Alias('P2')][int]$Paramtwo
       )
    <# Code Goes Here #>
    }
```

Mandatory
Position
HelpMessage
ValueFromPipeline
ValueFromPipelineByPropertyName
ValueFromRemainingArguments
DontShow
ParameterSetName

PoSh Parameter Validation

```
<#...#>
Function Invoke-DoStuff{
    [Alias('DoStuff')]
    Param(
       # This is Param One Description
       [ValidateSet('ValueA','ValueB')]
       [Parameter()][Alias('P1')][string]$ParamOne,
       # This is Param Two Description
       [ValidateRange(0,100)]
       [Parameter()][Alias('P2')][int]$Paramtwo
       )
    <# Code Goes Here #>
    }
```

PoSh Parameter Validation

```
<#...#>
Function Invoke-DoStuff{
    [Alias('DoStuff')]
    Param(
       # This is Param One Description
       [ValidateSet('ValueA','ValueB')]
       [Parameter()][Alias('P1')][string]$ParamOne,
       # This is Param Two Description
       [ValidateRange(0,100)]
       [Parameter()][Alias('P2')][int]$Paramtwo
       )
    <# Code Goes Here #>
    }
```

ValidateSet
ValidateLength
ValidateCount
ValidateRange
ValidatePattern
ValidateScript
ValidateNotNull
ValidateNotNullOrEmpty
AllowNull
AllowEmptyString
AllowEmptyCollection

PoSh Cmdlet Bindings

```
<#...#>
Function Invoke-DoStuff{
    [CmdletBinding()]
    [Alias('DoStuff')]
    Param(
       # This is Param One Description
       [ValidateSet('ValueA','ValueB')]
       [Parameter()][Alias('P1')][string]$ParamOne,
       # This is Param Two Description
       [ValidateRange(0,100)]
       [Parameter()][Alias('P2')][int]$Paramtwo
       )
    <# Code Goes Here #>
    }
```

PoSh Cmdlet Bindings

```
<#...#>
Function Invoke-DoStuff{
    [CmdletBinding()]
    [Alias('DoStuff')]
    Param(
       # This is Param One Description
       [ValidateSet('ValueA','ValueB')]
       [Parameter()][Alias('P1')][string]$ParamOne,
       # This is Param Two Description
       [ValidateRange(0,100)]
       [Parameter()][Alias('P2')][int]$Paramtwo
       )
    <# Code Goes Here #>
    }
```

Verbose
Debug
ErrorAction
Warning Action
InformationAction
ErrorVariable
InformationVariable
OutVariable
OutBuffer
PipelineVariable

PoSh Cmdlet Bindings

```
<#...#>
Function Invoke-DoStuff{
    [CmdletBinding()]
    [Alias('DoStuff')]
    Param(
       # This is Param One Description
       [ValidateSet('ValueA','ValueB')]
       [Parameter()][Alias('P1')][string]$ParamOne,
       # This is Param Two Description
       [ValidateRange(0,100)]
       [Parameter()][Alias('P2')][int]$Paramtwo
       )
    <# Code Goes Here #>
    }
```

HelpURI
ConfirmImpact
PositionalBinding
SupportsPaging
SupportsShouldProcess
DefaultParameterSetName

PoSh

Output Type

```
<#...#>
Function Invoke-DoStuff{
    [CmdletBinding()]
    [Alias('DoStuff')]
    [OutputType()]
    Param(
       # This is Param One Description
       [ValidateSet('ValueA','ValueB')]
       [Parameter()][Alias('P1')][string]$ParamOne,
       # This is Param Two Description
       [ValidateRange(0,100)]
       [Parameter()][Alias('P2')][int]$Paramtwo
       )
    <# Code Goes Here #>
    }
```

Final Touch…

PoSh

Voila !

```
<#...#>
Function Invoke-DoStuff{
    [CmdletBinding()]
    [Alias('DoStuff')]
    [OutputType()]
    Param(
       # This is Param One Description
       [ValidateSet('ValueA','ValueB')]
       [Parameter()][Alias('P1')][string]$ParamOne,
       # This is Param Two Description
       [ValidateRange(0,100)]
       [Parameter()][Alias('P2')][int]$Paramtwo
       )
    <# Code Goes Here #>
    }
```

PoSh

Before

```
Function DoStuff{
    <# Code Goes Here #>
    }
```

PoSh

After

```
<#...#>
Function Invoke-DoStuff{
    [CmdletBinding()]
    [Alias('DoStuff')]
    [OutputType()]
    Param(
       # This is Param One Description
       [ValidateSet('ValueA','ValueB')]
       [Parameter()][Alias('P1')][string]$ParamOne,
       # This is Param Two Description
       [ValidateRange(0,100)]
       [Parameter()][Alias('P2')][int]$Paramtwo
       )
    <# Code Goes Here #>
    }
```

EXAMPLES

RTFM Express
Basic Recipe & More
Cmdlet Bindings
Parameter Validation
value from Pipeline

RTFM Express
Basic Recipe & More
Cmdlet Bindings
Parameter Validation
value from Pipeline

ISE

Graphs

RTFM Express

Parameter Sets

ParameterSets

Example1

```
Invoke-Example
```

- Function
- Param Mandatory
- Param Optional

ParameterSets

Example1

[String]

[Switch]

Function

Param Mandatory

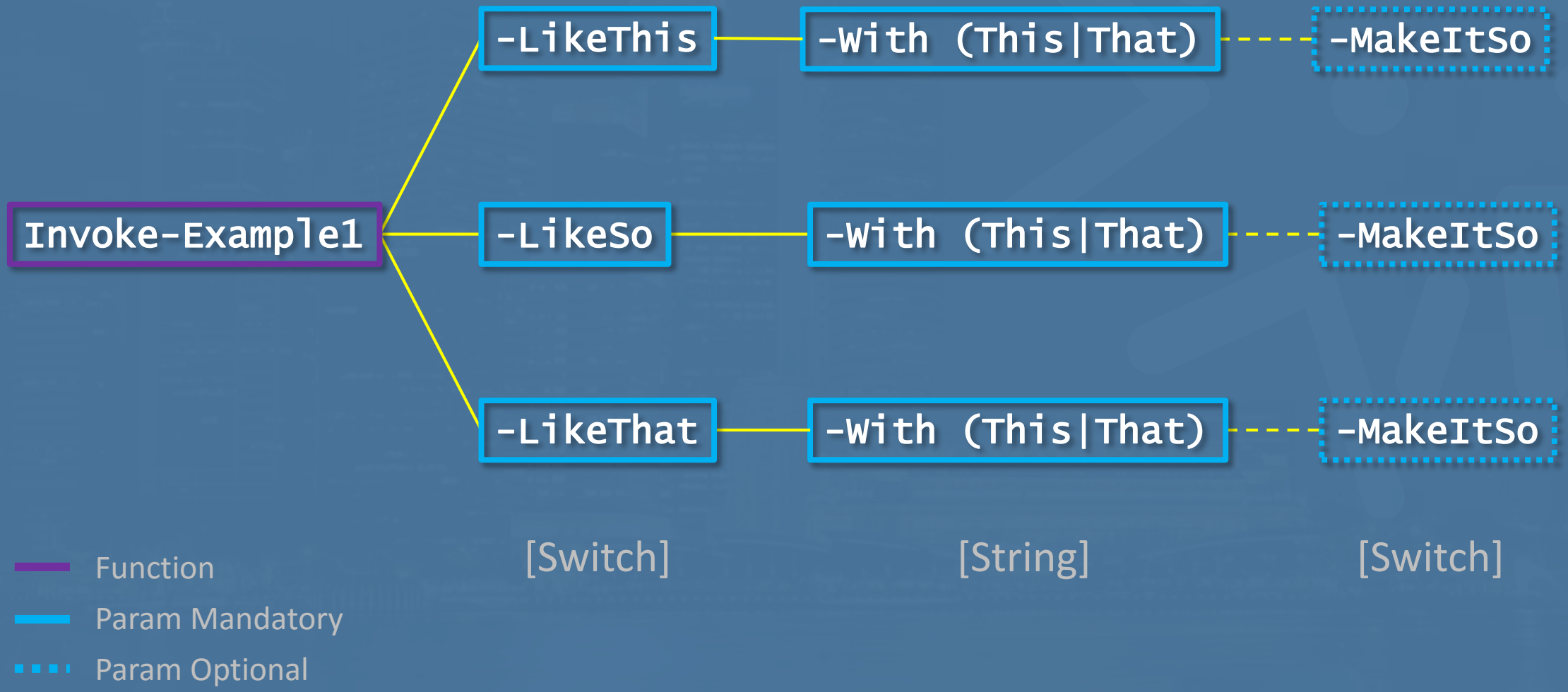
Param Optional

ParameterSets

Example1

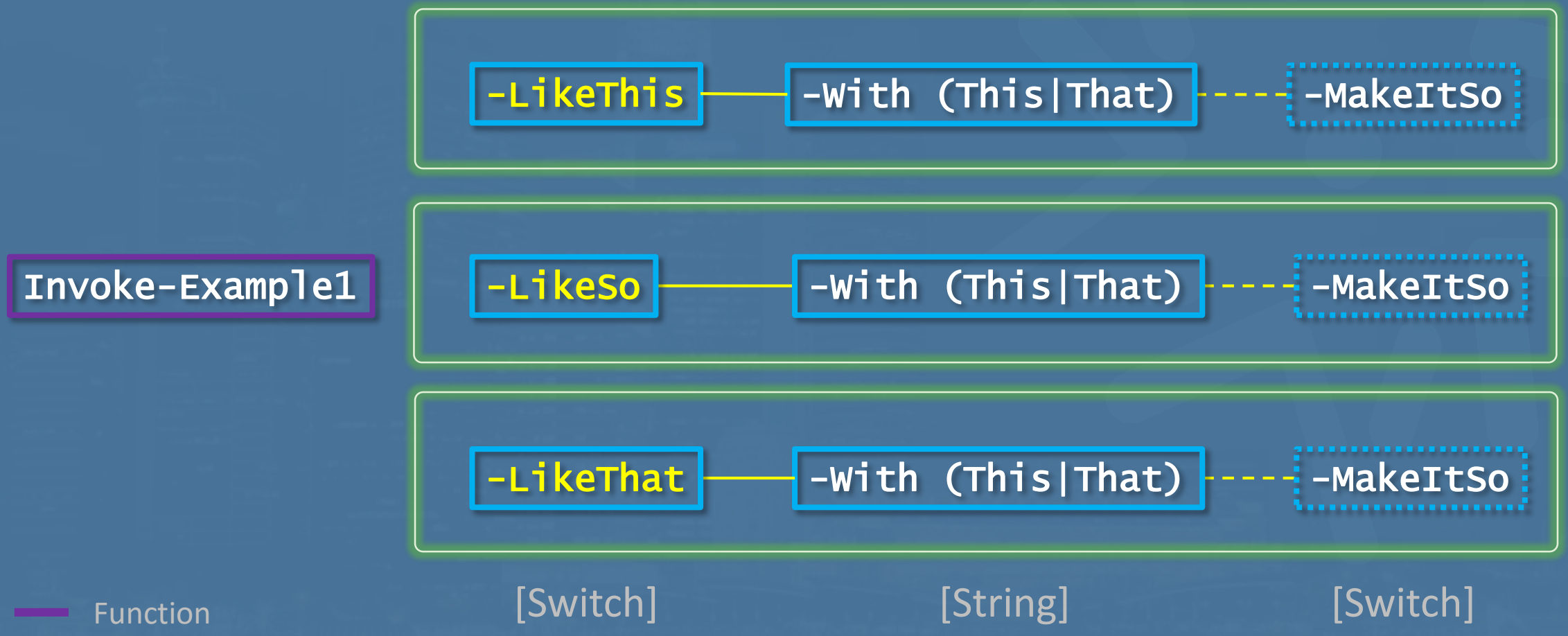
ParameterSets

Example1



ParameterSets

Example1



ParameterSets

Example1

```
Invoke-Example1    -LikeThis    -With <String>    [-MakeItSo]
```

```
Invoke-Example1    -LikeSo      -With <String>    [-MakeItSo]
```

```
Invoke-Example1    -LikeThat    -With <String>    [-MakeItSo]
```

Help Invoke-Example1

ParameterSets

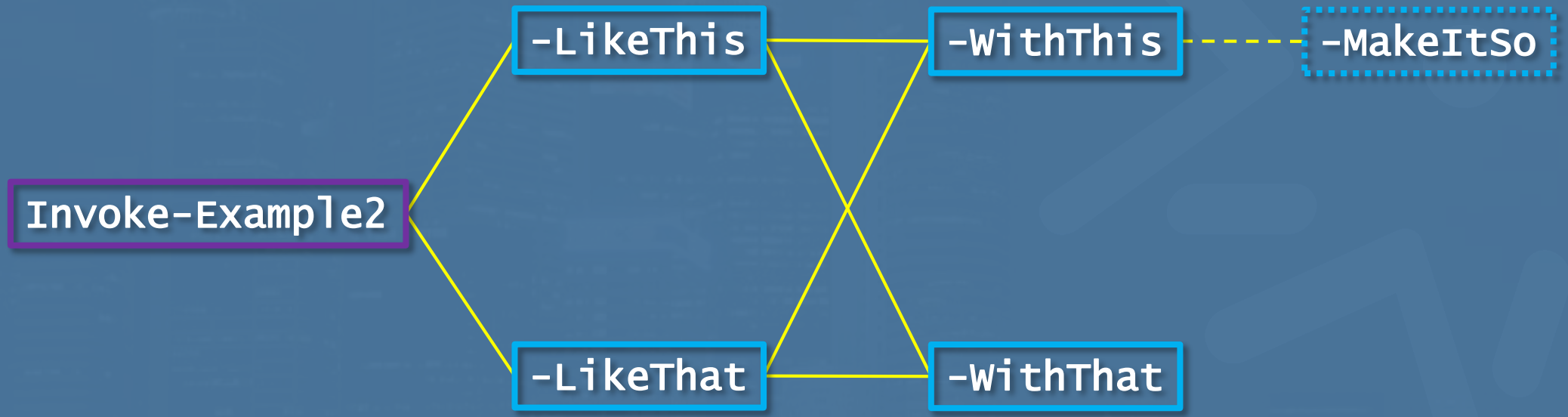
Example2

```
Invoke-Example2
```

- Function
- Param Mandatory
- Param Optional

ParameterSets

Example2

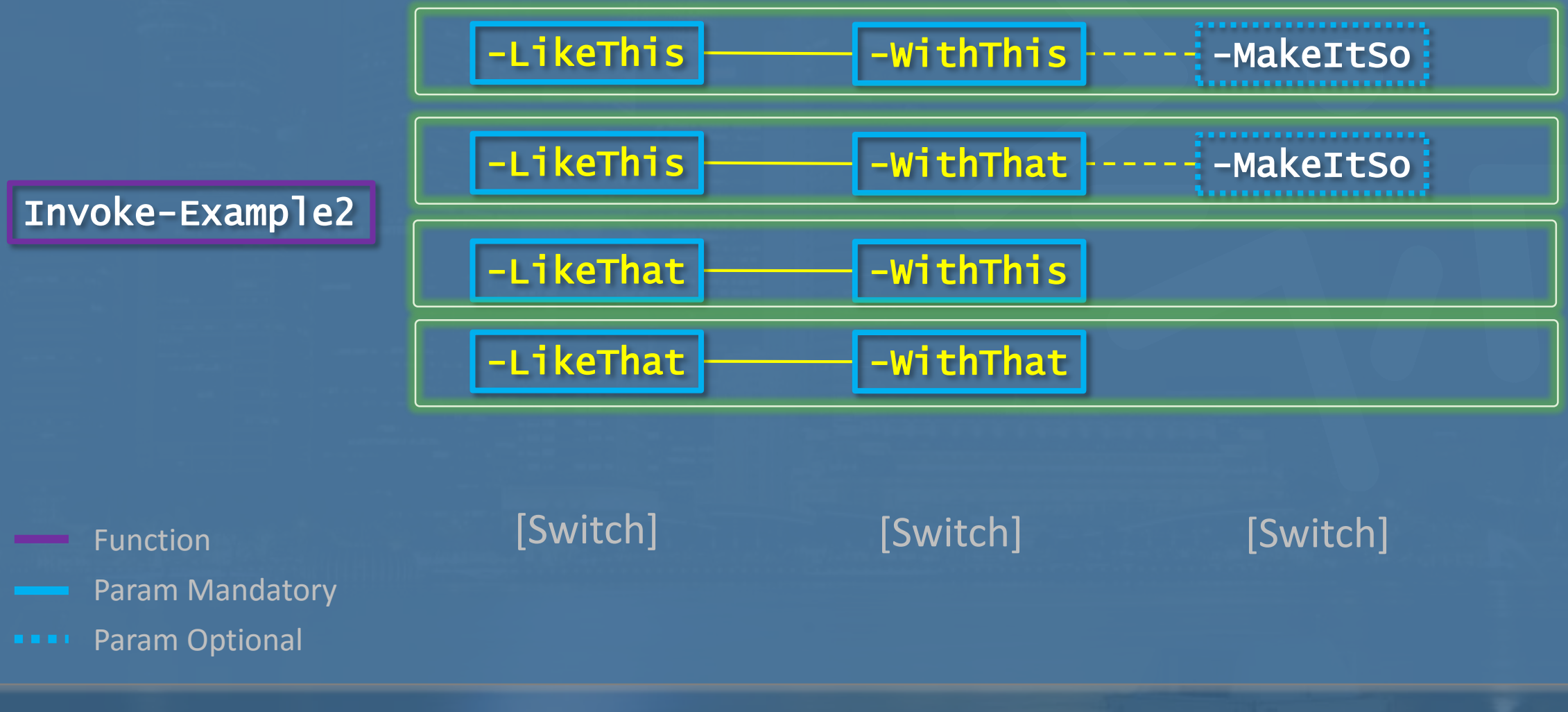


ParameterSets

Example2

ParameterSets

Example2



ParameterSets

Example2

```
Invoke-Example2     -LikeThis     -WithThis     [-MakeItSo]
Invoke-Example2     -LikeThis     -WithThat     [-MakeItSo]
Invoke-Example2     -LikeThat     -WithThis
Invoke-Example2     -LikeThat     -WithThat
```

Help Invoke-Example2

EXAMPLES

RTFM Express
Parameter Sets
Dynamic Parameters

RTFM Express
Parameter Sets
Dynamic Parameters

ISE

2 - SCRIPTING ON SCRIPTONITE

A Tool that make Tools


DEMO
ScriptMonkey

ScriptMonkey

ISE

3 - SECRETS OF THE DONUT MASTER

Writing Code
that writes Code
that writes Code

Don't Code!

SECRETS OF THE DONUT MASTER

Don't Code:

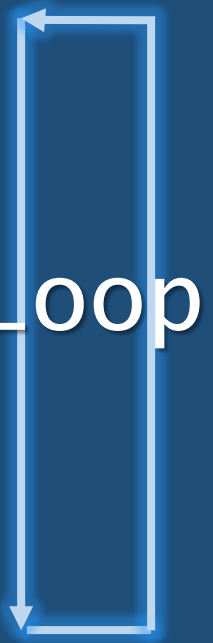
- Have a Donut
- Think…
- Graph…
- Re-Think…
- Re-Graph.

SECRETS OF THE DONUT MASTER

SECRETS OF THE DONUT MASTER

Don't Code:

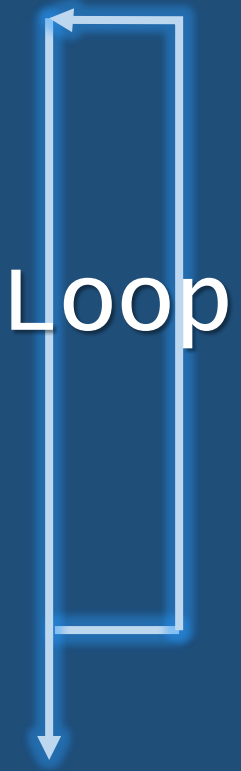
- Have a Donut
- Think…
- Graph…
- Re-Think…
- Re-Graph.



SECRETS OF THE DONUT MASTER

Don't Code:

- Have a Donut
- Think…
- Graph…
- Re-Think…
- Re-Graph.



Code skeleton > Test Syntax > Add Commands

DEMO

Automated ScriptMonkey
out of a CSV

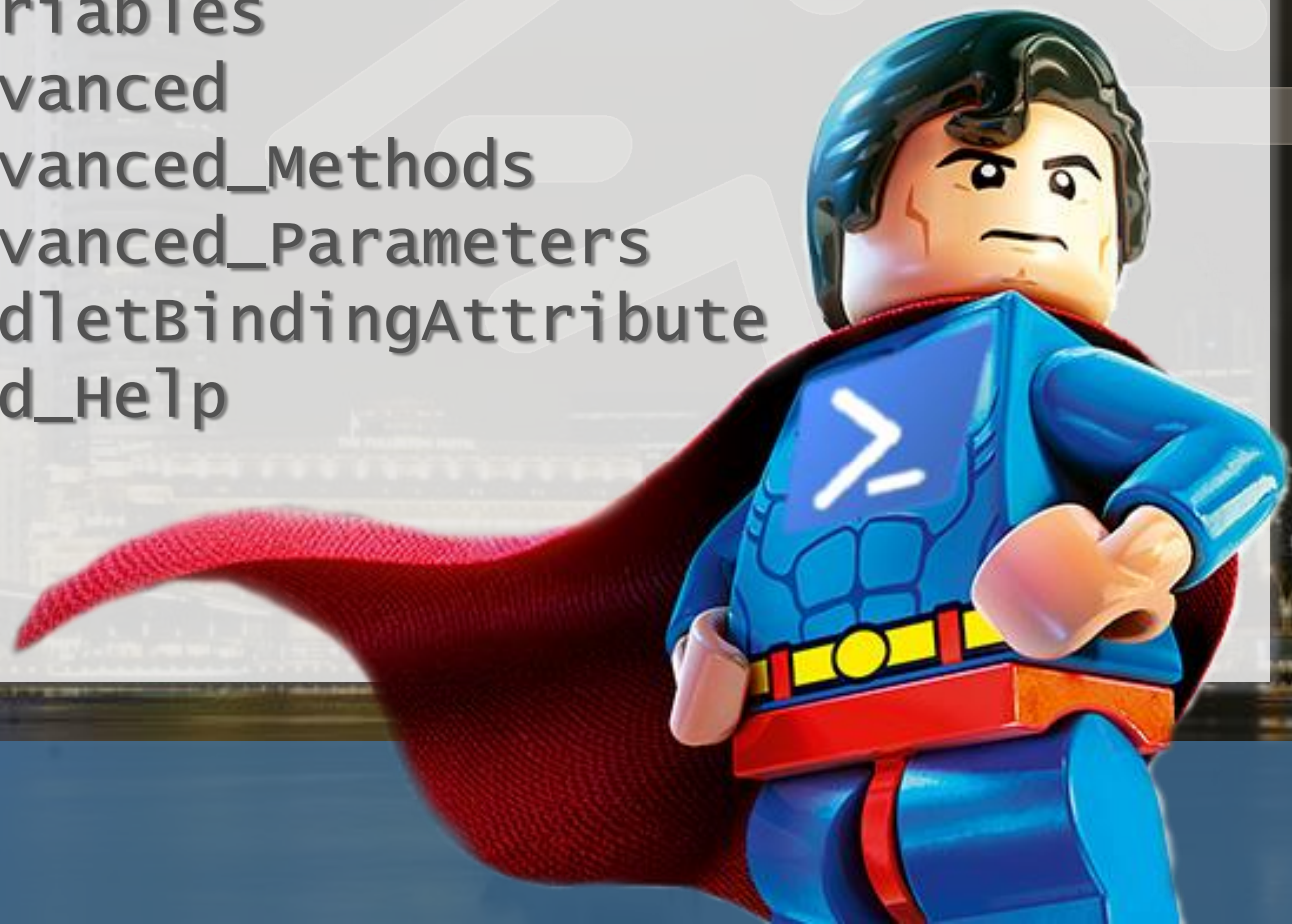
Automated ScriptMonkey out of a CSV

ISE

Get-Help


```
Get-Help about_Functions
Get-Help about_Parameters
Get-Help about_Automatic_Variables
Get-Help about_Functions_Advanced
Get-Help about_Functions_Advanced_Methods
Get-Help about_Functions_Advanced_Parameters
Get-Help about_Functions_CmdletBindingAttribute
Get-Help about_Comment_Based_Help
```

Also available online



Get-Blog

Boe Prox
https://learn-powershell.net/2014/02/04/using-powershell-parameter-validation-to-make-your-day-easier/

Mike Robbins
http://mikefrobbins.com/2015/03/31/powershell-advanced-functions-can-we-build-them-better-with-parameter-validation-yes-we-can/

FX Cat
http://www.lazywinadmin.com/2015/03/standard-and-advanced-powershell.html

Matt Graeber
http://www.powershellmagazine.com/2013/12/09/secure-parameter-validation-in-powershell/

Tobias
http://www.powertheshell.com/bp_function_parameters/

Trevor Sullivan
https://trevorsullivan.net/2010/11/15/powershell-dynamic-parameters-and-parameter-validation

FoxDeploy
https://foxdeploy.com/2017/01/13/adding-tab-completion-to-your-powershell-functions/

Thank You

@You -> For attending
@PSConf_Asia -> For organizing the event
@Jaap_Brasser -> For pushing me to submit Talks
@jsnover -> For PowerShell
MVA JumpStart Changed my Life…

@MyWife -> For doing all the rest
while I do this…

PowerShell Conference
Singapore 2017

Make IT So…

PowerShell Conference
Singapore 2017

Microsoft

KEMP
APPLICATION DELIVERY

SAPIEN
Technologies, Inc.

CHEF
CHEF.IO

Make IT So…

PowerShell Conference
Singapore 2017