PowerShell

SYED AWASE KHIRNI

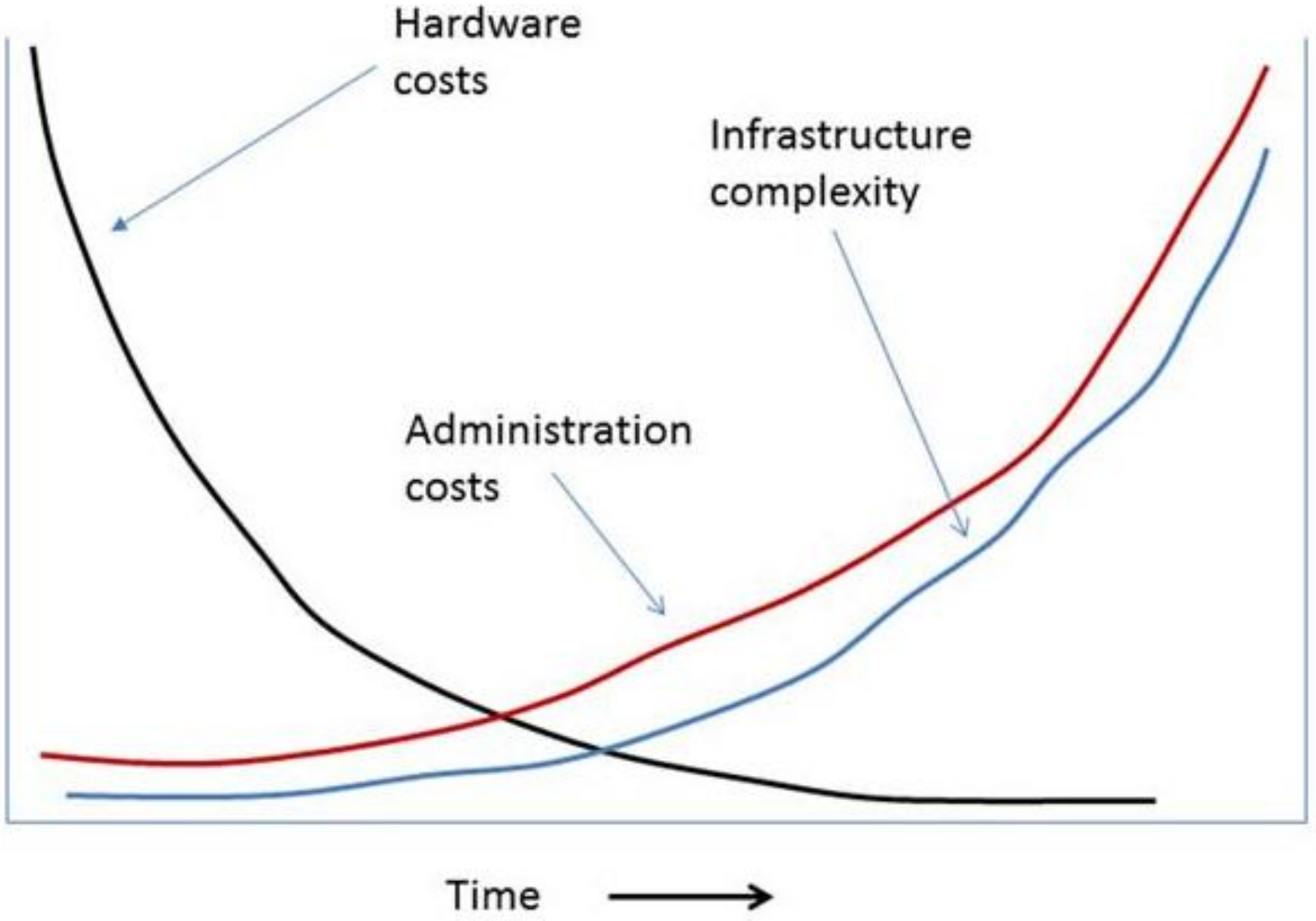
RESEARCHER | ENTHREPRENEUR | TECHNOLOGY COACH

awasekhirni@gmail.com | @aks008 | +91.9035433124

Syed Awase earned his PhD from University of Zurich in GIS, supported by EU V Framework Scholarship from SPIRIT Project (www.geo-spirit.org). He currently provides consulting services through his startup www.territorialprescience.com and www.sycliq.com. He empowers the ecosystem by sharing his technical skills worldwide, since 2008.

Terms of use

- You shall not circulate these slides without written permission from Territorial Prescience Research I Pvt ltd.
- If you use any material, graphics or code or notes from these slides, you shall acknowledge the author Dr. Syed Awase Khirni
- If you have not received this material, post-training session, you shall destroy it immediately and not use it for unauthorized usage of the material.
- Powershell is a copyright of Microsoft



PowerShell?

https://msdn.microsoft.com/en-us/powershell/scripting/powershell-scripting

- A terminal for managing windows infrastructure using scripting /commands to automate processes and workflows.
- Robust, Object oriented scripting language with access to a wide range of things on the computer
  - Access to entire .NET and WMI frameworks
- Enables administrators to perform administrative tasks on both local and remote windows systems.
- Automating various administrative tasks and processes
- Similar to Microsoft Management Console (MMC)
- Task Automation and configuration management framework
- Consisting of a command-line shell and associated scripting language
- Built on the .NET Framework
- Provides full access to COM and WMI

Using PowerShell

- PowerShell
  - Console
  - ISE: Integrated Scripting Environment
- Third-party Powershell editors
  - PrimalScript
  - PowerGUI
  - PowerShell +
  - PowerSE

- Powershell available for
  - Windows 8.1 x86 and x64
  - Windows Server 2012 R2 x64
  - Windows Management Framework (WMF)
    - Windows 7 SPI (or above) x86 and x64
    - Windows Embedded Standard 7
    - Windows Server 2008 R2 SPI (or above) x64
    - Windows Server 2012

PowerShell? Before and Now?

	Before	Now
GUI	MMC	GUIs built on top of Powershell
Interactive Shell	CMD	PowerShell
Scripting	BAT in CMD	PowerShell
COM	WMI(VBScript and Jscript)	PowerShell

PowerShell Compatibility

	PS1.0	PS2.0	PS3.0	PS4.0	PS5.0
Windows XP	SP2+	SP3+	Incompatible	Incompatible	
Server 2003	SP2+	SP2+	Incompatible	Incompatible	
Windows Vista	Installed	SP1+	Incompatible	Incompatible	
Server 2008	Installed	SP1+	SP2+	Incompatible	
Windows 7	Incompatible	Installed	Compatible	Compatible	
Server 2008 R2	Incompatible	Installed	SP1+	SP1+	
Windows 8	Incompatible	Version parameter	Installed	SP1+	
Server 2012	Incompatible	Version parameter	installed	Compatible	
Windows 8.1	Incompatible	Version parameter	Version parameter	Installed	
Server 2012 R2	Incompatible	Version parameter	Version parameter	Installed	
Windows 10					Compatible

PowerShell Major Changes

Version 2.0

- PowerShell Remoting
- New PowerShell ISE
- PowerShell Modules
- Background Jobs
- Many new Cmdlets
- Invoke-Command

Version 3.0

- Overhauled PowerShell ISE
- PowerShell Workflow
- Disconnected Sessions
- Scheduled Jobs
- Delegated Administrations
- Cmdlet Discovery
- Show-Command
- Invoke-WebRequest

PowerShell Major Changes

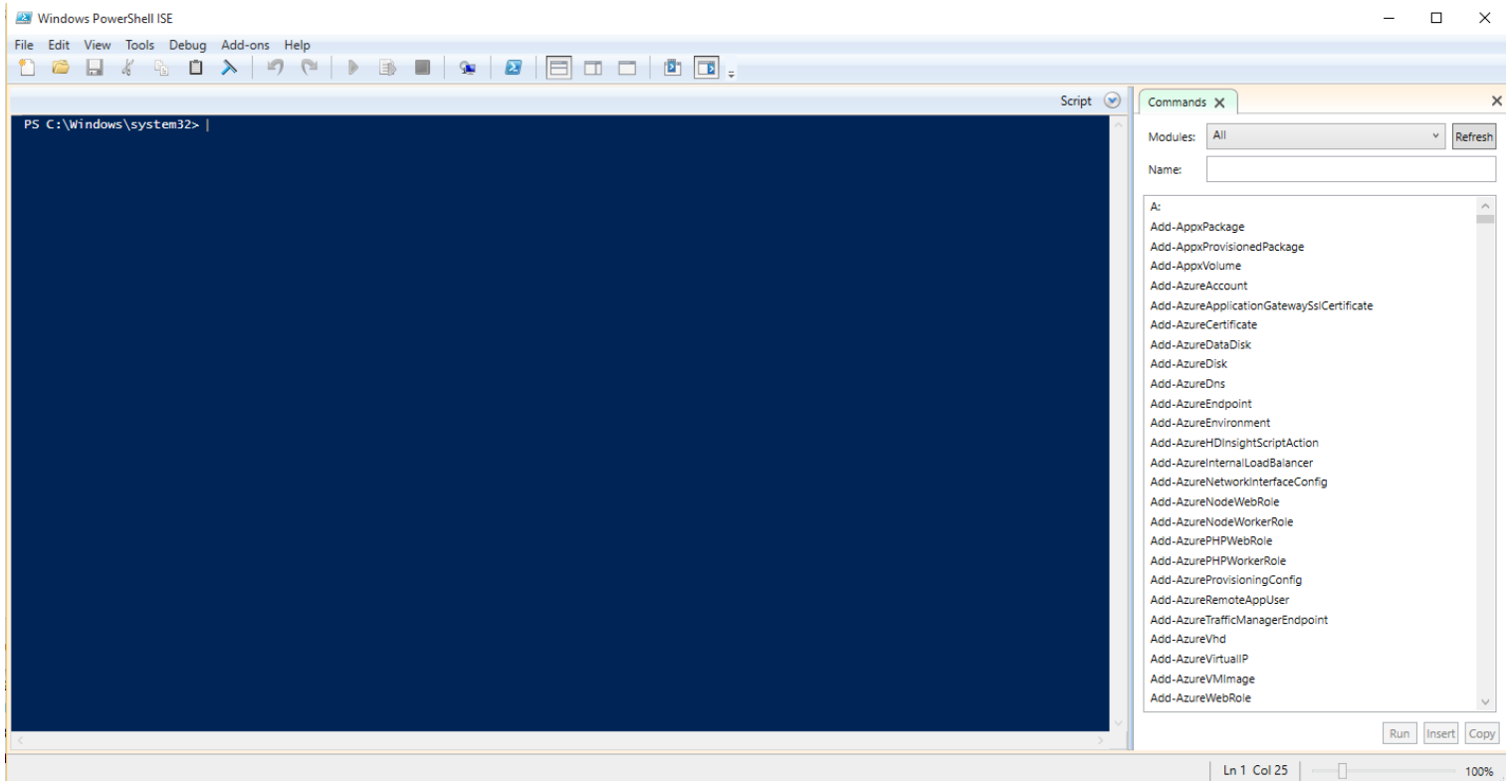
Version 4.0

- Improved Debugging
- Desired State Configuration Cmdlet's

Version 5.0

PowerShell –Integrated Scripting Environment

$PSVersionTable - to get the version



- configuration files (.ps1xml),
- module script files (.psm1), and
- Windows PowerShell profiles (.ps1).

Powershell References

- https://msdn.microsoft.com/en-us/library/dd835506(v=vs.85).aspx
- https://msdn.microsoft.com/en-us/library/ff458115.aspx
- https://msdn.microsoft.com/en-us/library/dd878310(v=vs.85).aspx
- https://msdn.microsoft.com/en-us/library/dd878294(v=vs.85).aspx
- https://msdn.microsoft.com/en-us/library/ee126192(v=vs.85).aspx
- https://msdn.microsoft.com/en-us/library/ee706563(v=vs.85).aspx
- https://msdn.microsoft.com/en-us/library/gg580944(v=vs.85).aspx
- https://msdn.microsoft.com/en-us/library/ms714469(v=vs.85).aspx

- Get-Command
- Get-Help
- Get-PSDrive

PowerShell Execution Policies

https://msdn.microsoft.com/powershell/reference/5.1/Microsoft.PowerShell.Core/about/about_Execution_Policies

- execution policies let you determine the conditions under which Windows PowerShell loads configuration files and runs scripts.
- set an execution policy for the local computer, for the current user, or for a particular session.
- Execution policies for the local computer and current user are stored in the registry.

```
Get-ExecutionPolicy
Get-ExecutionPolicy -List
Get-ExecutionPolicy -Scope CurrentUser
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned
Set-ExecutionPolicy RemoteSigned -Scope CurrentUser
```

- Execution Policies
  - Restricted - permits individual commands, but will not run scripts
  - prevents running of all scritps

Restricted Execution Policy

- Default execution policy in Windows 8, Windows Server 2012, and Windows 8.1.
- Permits individual commands, but will not run scripts.
- Prevents running of all script files, including formatting and configuration files (.ps1xml), module script files (.psm1), and Windows PowerShell profiles (.ps1).

AllSigned Execution Policy

- Scripts can run.
- Requires that all scripts and configuration files be signed by a trusted publisher, including scripts that you write on the local computer.
- Prompts you before running scripts from publishers that you have not yet classified as trusted or untrusted.
- Risks running signed, but malicious, scripts.

Remote Signed Execution Policy

- Scripts can run. This is the default execution policy in Windows Server 2012 R2.
- Requires a digital signature from a trusted publisher on scripts and configuration files that are downloaded from the Internet (including e-mail and instant messaging programs).
- Does not require digital signatures on scripts that you have written on the local computer (not downloaded from the Internet).
- Runs scripts that are downloaded from the Internet and not signed, if the scripts are unblocked, such as by using the Unblock-File cmdlet.
- Risks running unsigned scripts from sources other than the Internet and signed, but malicious, scripts.

Unrestricted Execution Policy

- Unsigned scripts can run. (This risks running malicious scripts.)
- Warns the user before running srcipts and configuration files that are downloaded from the Internet.

Bypass Execution Policy

- Nothing is blocked and there are no warnings or prompts.
- This execution policy is designed for configurations in which a Windows PowerShell script is built in to a a larger application or for configurations in which Windows PowerShell is the foundation for a program that has its own security model.

Undefined Execution Policy

- There is no execution policy set in the current scope.
- If the execution policy in all scopes is Undefined, the effective execution policy is Restricted, which is the default execution policy.

Vocabulary

- String: Any combination of letters and numbers “asdad”, “ab123”
- Integer – A number without quotes, used for math
- Boolean – True or False, represented in PS as $True or $False
- Variable : A reference to a value that can be assigned over the course of a script/program. Declared in Powershell with a **$[word]**

example : $variable1

Objects

- An object is a type of “something”
- As an object of a specific type, it inherits properties and methods related to it’s object type
  - Properties – information about the object
  - Methods – Code that interacts with the object

- $string.Length
- $string.ToUpper()

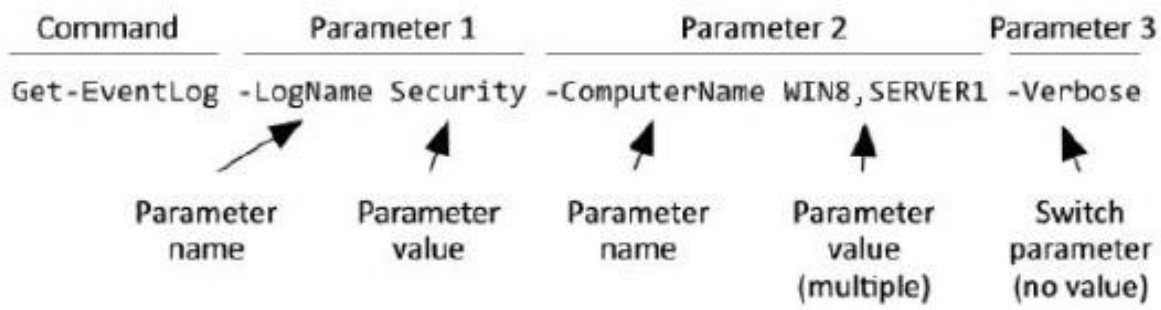
Arrays

- A list of objects separated by commas
- ex: “one”,”two”,”three”
- Arrays can be accessed using index numbers

Commands

- Powershell has 4 features that we think of as commands
  - Internal cmdlets, which only run inside PowerShell and are written in a .NET framework language such as Visual Basic or C#
  - Functions, which are written in PowerShell's scripting language
  - PowerShell v3 and v4 cmdlets, which are produced from WMI (Windows Mangement Instrumentations) classes using "cmdlets over objects" capabilities.

- External Commands such as ping.exe which could also be run from old cmd.exe shell.



CMDLETS

- "Writing a Windows PowerShell Cmdlet" is for program managers who are designing cmdlets and for developers who are implementing cmdlet code.
- A lightweight command that is used in the Windows PowerShell environment.
- Windows PowerShell runtime invokes these cmdlets within the context of automation scripts that are provided at the command line.
- Cmdlets perform an action and typically return a Microsoft .NET Framework object to the next command in the pipeline.
- Primary way of getting things done in PowerShell
- Always in a "**verb-noun**" format
  - Write-host – print something to screen
  - Get-process – get running processes
  - Set-clipboard – copy something to clipboard
  - get-eventlog – get contents of eventlog

Cmdlet

- Name
- Synopsis
- Syntax
- Description
- Parameters
- Inputs
- Outputs
- Notes

- Parameter Sets
  - Cmdlet's can have multiple sets
  - Each set has similar, but discrete parameter functionality
  - Each set can share parameters
- Positional Parameters
  - Able to execute command without specifying name
  - Value must be in correct order
  - Get-Command *VM ComputerName

```
Get-Random[[-Maximum] <Object>] [-Minimum <Object>][-SetSeed <int>][<CommonParameters>]
```

Cmdlet

- Named Parameters
  - Requires parameter name be specified prior to value
  - Does not have be in specific order
- Mandatory Parameters
  - Most cmdlet sets require atleast one parameter
- Optional Parameters
  - A parameter set normally includes optional parameters
  - Identified by [] brackets

```
Get-Command -ArgumentList ComputerName -Name *VM
```

Script blocks

- A script block can contain any set of PowerShell commands and it can contain as many of them as you need.

```
$sb={
Get-CimInstance -ClassName Win32_OperatingSystem
Get-CimInstance -ClassName Win32_ComputerSystem
}
```

Logical Operators

	Lt	Less than
	Le	Less than or equal to
	Eq	Equal to
	Ne	Not equal to
	Ge	Greater than or equal to
	Gt	Greater than

Get-Help

- Show general information on using help
  - Get-Help
- Show help for specific command
  - Get-Help Get-Command
- Show in-depth details for specific command
  - Get-Help Get-Command –Full
- Show examples for a specific command
  - Get-Help Get-Command –Examples
- Help Categories
  - Alias, Cmdlet, Function, Provider,Workflow, HelpFile

PowerShell Extensions

- PowerShell Snapins
  - Depreciated
  - .NET assembly that implements cmdlets and PS Providers
- PowerShell Modules
  - A package of cmdlets that you can use in PowerShell

- PowerShell Snapins
  - Microsoft SQL Server 2008 R2
    - SqlServerCmdletSnapin100
    - SqlServerProviderSnapin100
  - Microsoft SharePoint Server
    - Microsoft.SharePoint.PowerShell
  - Vmware PowerCLI
    - Package of 7 different Snapins

PowerShell Modules

- Community Built
  - Hundreds of Modules available on TechNet Gallery
- Packaged with OS
  - ServerManager
  - BestPractices
  - Many more
- Additional Installation
  - SQL Server 2012 R2

PowerShell Provider?

•They are Microsoft .NET Framework based programs that make the data in a specialized data store available in Windows PowerShell so that you can view and manage it

- Default PowerShell Providers
  - •Alias
  - •Environment
  - •FileSystem
  - •Function
  - •Registry
  - •Variable
  - •Certificate
  - •WSMAN
  - •SqlServer(other)
  - •VMWare (other)

PowerShell Drive

- A data store location that you can access like a file system drive in Windows PowerShell.
- Windows PowerShell providers create some drives for you, such as the file system drives, the registry drives and the certificate drive and also users can create their own Windows PowerShell drives.

- Default PowerShell Drives
  - Alias
  - C(and any other disk drive)
  - Cert
  - Env
  - Function
  - HKCU (Registry drive)
  - HKLM (Registry)
  - Variable
  - WSMAN

```
5  Get-PSDrive
6
7  //create a new fileSystem PSDrive
8  New-PSDrive -Name S -PSProvider FileSystem -Root \\Server01\\SyedAwase
```

```
//Remove a PSDrive
Remove-PSDrive -Name S
```

PSDrives and PSProviders

- It is a data store location that you can access like a file system drive in Windows Powershell.
- It uses the noun, PSDrive for the commands that work with windows powershell drive.
- PSProvider parameter lets you display only the Windows PowerShell drives that are supported by a particular provider.

```
AKS>>>Get-PSProvider

Name                 Capabilities                    Drives
----                 ------------                    ------
Registry             ShouldProcess, Transactions     {HKLM, HKCU}
Alias                ShouldProcess                   {Alias}
Environment          ShouldProcess                   {Env}
FileSystem           Filter, ShouldProcess, Cre...   {C, E, F}
Function             ShouldProcess                   {Function}
Variable             ShouldProcess                   {Variable}
Certificate          ShouldProcess                   {Cert}
WSMan                Credentials                     {WSMan}
```

```
PS C:\Windows\system32> Get-PSDrive -PSProvider FileSystem

Name           Used (GB)     Free (GB) Provider      Root
----           ---------     --------- --------      ----
C                 154.29        116.16 FileSystem    C:\
D                3675.49         50.41 FileSystem    D:\
E                 180.45         14.37 FileSystem    E:\
F                                       FileSystem    F:\
K                                       FileSystem    K:\
```

```
PS C:\Windows\system32> Get-PSDrive -PSProvider Registry

Name           Used (GB)     Free (GB) Provider      Root
----           ---------     --------- --------      ----
HKCU                                   Registry      HKEY_CURRENT_USER
HKLM                                   Registry      HKEY_LOCAL_MACHINE
```

Cmdlets

https://technet.microsoft.com/en-us/library/hh848794.aspx

- They are heart-and-soul of windows PowerShell, Microsoft's new command shell/scripting language.

• Dates and Times • Files and Folders • Help and Information • Saving and Importing Data • Scripting Techniques • Scripts and Applications	• Security and Security Descriptors • System Administration Tasks • Windows PowerShell Aliases • Windows PowerShell Drives and Namespaces • Windows PowerShell Sessions

Item Cmdlets

	Cmdlet	Action
	Clear-Item	Remove the value of an item
	Copy-Item	Copy an item to a new location
	Get-Item	Retrieve data from an item
	Get-ChildItem	Retrieve child items from parent
	Invoke-Item	Performs action for a specified item
	Move-Item	Moves an item to a new location
	New-Item	Creates a new item
	Remove-Item	Deletes an item
	Rename-Item	Renames an item
	Set-Item	Updates the value of an item

Item Property Cmdlets

	CmdLet	Action
	Clear-ItemProperty	Delete value of a property
	Copy-ItemProperty	Copy property from one location to another
	Get-ItemProperty	Retrieve properties of item
	Move-ItemProperty	Move a property and it's values from one locations to another
	New-ItemProperty	Create new item Property
	Remove-ItemProperty	Delete a property and it's values
	Rename-ItemProperty	Rename a property
	Set-ItemProperty	Update the properties of an item

Location Cmdlets

	Cmdlet	Action
	Get-Location	Retrieve current working location
	Pop-Location	Change current location to most recently pushed
	Push-Location	Adds current location to the top of a list of locations
	Set-Location	Set current working location

Path Cmdlets

	Cmdlet	Action
	Join-Path	Combine a parent and child path segment to create a provider-internal path
	Convert-Path	Convert a path from a PowerShell path to a PowerShell provider path
	Split-Path	Returns a specified part of a path
	Resolve-Path	Resolves the wildcard characters in a path
	Test-Path	Determines if all elements of a path exist

Service Cmdlets

	Cmdlet	Action
	Get-Service	Returns all of the services for the computer
	New-Service	Creates a New Service in the service database
	Restart-Service	Restarts a specified service
	Resume-Service	Resumes a specified service from a suspended state
	Set-Service	Modify the state or the property of a specified service
	Start-Service	Start a specified service
	Stop-Service	Stop a specified service
	Suspend-Service	Suspend a specified service

Managing Process Cmdlets (Not Task Manager)

	Cmdlet	Action
	Debug-Process	Attaches debugger to specified process(s)
	Get-Process	Returns all processes running
	Start-Process	Starts a new process
	Stop-Process	Stops a process
	Wait-Process	Suppresses the command prompt until a process is stopped

EventLog Cmdlets

	Cmdlet	Action
	Clear-EventLog	Deletes event log Entries from specified event log
	Get-EventLog	Retrieves events from specified event log
	Limit-EventLog	Sets maximum size of event log you specify
	New-EventLog	Create new event log
	Remove-EventLog	Deletes an event log
	Show-EventLog	Opens event viewer
	Write-EventLog	Write an event to a specified event log

PhysicalDisk Cmdlets

	Cmdlet	Action
	Add-PhysicalDisk	Adds specified disk to a specified storage pool or virtual disk
	Get-PhysicalDisk	Retrieves all physical disk(s)
	Remove-PhysicalDisk	Remove specified physical disk from specified storage pool
	Reset-PhysicalDisk	Reset status of a physical disk in a storage space
	Set-PhysicalDisk	Modifies attributes of a specified physical disk

Volume Cmdlets

	Cmdlet	Action
	Format-Volume	Format specified volumes
	Get-Volume	Retrieves all disk volumes
	New-Volume	Creates a new disk volume for a storage space
	Optimize-Volume	Trims, Defragments or consolidates slabs for specified volume
	Repair-Volume	Repairs Specified Volume
	Set-Volume	Modify the specified volume

WindowsFeature Cmdlets

	Cmdlet	Action
	Get-WindowsFeature	Retrieves Information about Windows Server roles. Role services and features
	Install-WindowsFeature	Installs the specified roles, role services and features
	Uninstall-WindowsFeature	Uninstalls and optionally removes specified roles, role services and features

PowerShell Pipeline

SYED AWASE

Pipeline

- A series of commands, where in the results of the preceding expression/command are sent as input to next command/expression using "pipe" **(|)** operator.
- You can identify if **pipeline input is accepted by looking at Code options using**
  - **Get-Help Get-Command -Full**

Measure-Object

- Calculate numeric properties of an object
- used to compute the minimum, maximum, Average and Sum of a property
- Counting Total Files in a Directory
  - Get-ChildItem –Path c:\Windows –File | Measure-Object
- Counting Lines in a text file
  - Get-Content C:\Syed\profile.txt | Measure-Object -Line
- Summing Total Disk Utilization
  - Get-ChildItem C:\Syed –Recurse –File | Measure-Object -Property Length -Sum

Measure-Object

```
Get-ChildItem -Path . -File
Get-ChildItem -Path . -File |Measure-Object
Get-ChildItem -Path . -File |Measure-Object -Property Length -Sum
$diskUsage = Get-ChildItem . -Recurse -File | Measure-Object -Property Length -Sum
$diskUsage.Sum/1GB
Get-Content .\testscore.txt
Get-Content .\testscore.txt | Measure-Object -Sum -Average -Minimum -Maximum
```

The Out Cmdlet

	Cmdlet	Action
	Out-File	Outputs the specified content to a file
	Out-GridView	Outputs the specified content to an interactive table
	Out-Host	Outputs the content to the console
	Out-Null	Deletes the content that would normally be output
	Out-Printer	Sends the output to a specified printer
	Out-String	Converts the output into a string
	Out-default	Ouput is directly sent to the host/the shell.

Convert Command

- Export in Powershell terms means converting data into a format say CSV, then saving the converted format into a file for storage
- Convert on the other hand means conversion to a different format only not necessarily saving it into a file
- Get-Service |ConvertTo-HTML | Out-file services.html

The ConvertTo Cmdlet

	Cmdlet	Action
	ConvertTo-Csv	Converts provided object to csv string
	ConvertTo-Html	Converts provided object to HTML string
	ConvertTo-Json	Converts provided object to JSON string
	ConvertTo-SecureString	Converts provided encrypted strings or plain text strings to a secure string
	ConvertTo-TpmOwnerAuth	Converts string value to a TPM owner authorization string
	ConvertTo-Xml	Converts provided object to an XML String
ConvertFrom		
	ConvertFrom-Csv	Converts a String in a csv form to an object
	ConvertFrom-JSON	Converts a JSON Object Structure to an object
	ConvertFrom-SecureString	Converts provided encrypted strings or plain text string to a secure string
	ConvertFrom-StringData	Converts a string that contains key and value pairs into a hash table

Import and Export XML Documents

- Exporting data to an XML document
  - Get-Service | Export-Clixml .\myservices.xml
- Importing data from an XML document
  - Import-Clixml .\myservices.xml

Importing and Exporting CSV Documents

- Exporting data to a CSV Document
  - Get-Service | Export-Csv .\myservices.csv
- Importing data from CSV Document
  - Import-Csv .\myservices.csv

Comparison Operators

	Operator	Definition
	-eq	Equal to
	-ne	Not equal to
	-gt	Greater than
	-lt	Less than
	-le	Less than equal to
	-ge	Greater than equal to
	-match	Matches a string using a regular expression
	-notmatch	Does not match a string using a regular expression
	-like	Match a value using a wildcard character
	-notlike	Does not match a value using a wild card character
	-In	Tells whether a test value appears in a collection or referenc evalue
	-notin	Tells whether a test value does not appear in a collection or reference value
	-contains	Determines elements in a group
	-notcontains	Determine excluded elements in a group

Boolean Operators

	Operator	Purpose
	-and	Return True if all subexpressions are TRUE
	-or	Return True if all subexpressions are TRUE
	-not or !	Return the opposite, True becomes false and viceversa
	-xor	Return True if one subexpression is TRUE, but not if both are True
	-replace	Replaces part of all of the value on the left side of operator

Bitwise Operators

	Operator	Purpose
	-band	Return 1 if both compared bits are 1
	-bor	Return 1 if either compared bit is 1
	-bxor	Return 1 if one compared bit is 1, but not if both are 1

Object Type Operators

- Powershell provides **three operators that identify and manipulate object types**
  - "**-is**" operator is used to test the type of an object
    - "Syed Awase" is [string]
  - "**-isnot**" performs the converse action – it tests if an object is not of a specific type
  - "**-as**" operator attempts to convert an object of one type into another type
    - "string" as [int]

Format Operator

- "-f" format operator to create formatted strings.

```
PS E:\CT\Powershell\CodePlay> "You have logged in on {0} and your credentials are {1}" -f (Get-Date),($Env:USERNAME)
You have logged in on 12/13/2016 10:47:40 AM and your credentials are SyedAwase
```

Call Operator

- **& sign (ampersand) – invoke operator** which is used **to execute strings or script blocks**
-

```
PS E:\CT\Powershell\CodePlay> $cmd = "dir"

PS E:\CT\Powershell\CodePlay> & $cmd
```

SubExpression Operator

- **$( )** is used to place expressions inside the parentheses
- **A subexpression enables you to have something more complicated than a single variable**

```
#subexpression
$service = Get-Service | select -First 1
"Service name is $($service.name)"
1..100 | foreach {Write-Host $_}
$procs = Get-Process
$procs[2..5]#range operator
```

Math Operators

- **Math class** has a number of methods that supply the math functionality needed for powershell operations.
  - [math]::pi
  - [System.Math]::pi
  - "**::**" tells PowerShell to use a static filed and pi determines the field or method to be accessed.
  - [math]::Sqrt(16)
  - [math]::Sin([math]::pi/2)

PowerShell

• PowerShell has two types of extensions: **PSSnapins** and **Modules**, Both are capable of adding **Cmdlets** and **PSProviders** to the shell.

PSSnapins

• They are first approach of extending the shell, although they are still supported in versions 2, 3, and 4.

•Deprecated now, but they are written in .NET language like C# or VB and they are packaged as DLL files.

Modules

•They are capable of adding functions to the shell, introduced in version 2.

•Modules can benefit from autoloading, they can also be copied from system to system and on their underlying dependencies.

Modules

- They are preferred way of extending the shell. **Using Profiles we can automate the loading of modules into the memory.**
- **Modules are novel way of extending the shell functionality.**

```
# show modules loaded by default
Get-Module
# display modules that are available to load
Get-Module -Name Cim* -ListAvailable
#importing CimCmdlets Module
# version 2, we had to use Import-Module CimCmdlets
Get-CimInstance win32_bios
#displaying the list of modules loaded now
Get-Module
#remove auto loaded module or any module |
Remove-Module CimCmdlets
```

Selecting Objects

• **Select-Object** includes a –Property parameter, which *accepts a comma-separated list of properties that you want to display.* We ***use this to override the default display for that object type.***

•Select-Object can do is ***select a subset of "rows" or objects.*** It can *select a chunk of objects either from the beginning of the set, or from the end, or even from the middle.*

```
#select-objects
Get-Process | Select-Object -Property Name, ID, VM, PM
Get-Process | Select Name,ID, VM, PM |Get-Member
Get-Process | Select Name, Id, PM, NPM | Sort VM-Descending
Get-Process |Sort VM-Descending| Select Name, Id, PM, NPM
#selecting object => subset of objects
Get-Process | sort VM -Descending | select -First 5
Get-Process | sort VM -Descending | select -Last 5
Get-Process | sort VM -Descending | select -skip 3 -First 5
Measure-Command {1..500 | select -First 5 -wait}
Measure-Command {1..5000 | select -First 5}
```

Making custom properties using select-object

- Select-Object's –Property parameter *accepts a combination of property names, and some custom properties which are properties* defined on the fly using a special syntax.

```
#making custom property => Select-Object
#creating a custom property "TotalMemory" => Computed Value
Get-Process | Select -Property Name, ID, @{name="TotalMemory"; expression={$_.PM+$_.VM}}
Get-Process | Select -Property Name, ID, @{name="TotalMemory"; expression={($_.PM+$_.VM) /1GB -as[int]}}
Get-Process |Select -Property Name, ID,
     @{Name="virtMem";Expression={$psitem.vm}},
     @{Name="PhysMem";Expression={$psitem.pm}}

Get-Process |Select -Property Name, ID,
     @{Name="virtMem";Expression={$psitem.vm}},
     @{Name="PhysMem";Expression={$psitem.pm}} | Get-Member
#choosing a subset of objects
Get-Process | sort PM -Descending | select -Skip 3 -First 5 -Property name, id, pm, vm
```

Filtering Objects (Where-Object)

- **Where-Object** is used for filtering out objects based on user's requirements.
- First introduced in PowerShell v3.0
- It requires
  - The name of the property that contains the data you want to filter on
  - The property values that you want to keep

```
#Filtering Objects using Where-Object
Get-Service | Where Status -ne Running
Get-Service | Where-Object Status -ne Running
Get-Service | ? Status -eq Running
Get-Service | Where status -eq stopped | where servicetype -eq Win32OwnProcess
Get-Process | Where-Object -FilterScript {$_.WorkingSet -gt 1mb -AND $_.company -notmatch "Microsoft"}
(Get-Process).Where({$_.ws -gt 100mb},"First",3)
(Get-Process).Where({$_.ws -gt 100mb},"Last",2)
Get-Process | Where {$_.ws -gt 100mb} | Select -First 3
measure-command {(1..1000) | where {$psitem%2}}
Get-Process | sort Handles
```

Grouping Objects

- **Group-Object** Cmdlet takes a bunch of object and puts them into buckets or groups based on a key property

```
#Grouping Object -------------------------

Get-Service | Group-Object -property Status
Get-Service | group status | Get-Member
$myservices = Get-Service | group status
$myservices
```

Enumerating Objects

```
#Enumerating Objects using ForEach-Object
Start-Process calc
Get-Process calc
Get-Process calc | foreach Kill
("Syed Awase","Syed Ameese","Syed Azeez", "TPRI","SycliQ").foreach({$_.toupper()})
Get-Process | Format-Wide
```

Special Characters

- Single Quotes will always handle a string quite literally
- **Double Quotes, on the other hand will allow you to escape the special characters that are in the text**
- **Special character, when preceded by the backtick, or ` will take on specific actions that otherwise would not be accomplished without the presence of the backtick.**

```
PS C:\Windows\system32> Write-Output (“`n`tSyed Awase” +
“`n`t`”escaped`” characters.`n”)


    Syed Awase
    ”escaped” characters.
```

Special Character	Description
`0	Insets a null value
`a	Sends an alert(bell or beep) to the computer's speaker
`b	Inserts a backspace
`f	Inserts a form feed
`n	Inserts a new line
`r	Inserts a carriage return
`t	Inserts a horizontal tab
`v	Inserts a vertical tab
`'	Inserts a single quote
`"	Inserts a double quote

Scalers

- They are virtual duffel bags that are utilized in order to store and transport data within your PS code.
- Scalers that are built into both PS's and Windows Environments come with data already allocated to them.

dir env: | sort name

```
PS C:\Windows\system32> $PSHOME
C:\Windows\System32\WindowsPowerShell\v1.0

PS C:\Windows\system32> dir $PSHOME -Filter *.dll


    Directory: C:\Windows\System32\WindowsPowerShell\v1.0


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---l         7/10/2015   4:29 PM          56320 PSEvents.dll
-a---l         7/10/2015   4:28 PM         174592 pspluginwkr.dll
-a---l         7/10/2015   4:29 PM           3072 pwrshmsg.dll
-a---l         7/10/2015   4:29 PM          29696 pwrshsip.dll
```

PS Providers

- the core of data store access resides within PS providers. One such providers is Microsoft .NET.
- It provides you with a data access layer. This layer is between the data and PS itself.
- This layer and by relation the providers, allow you to connect with various data stores by utilizing different PS mechanisms.

```
PS C:\Windows\system32> Get-PSProvider | Select Name

Name
----
Registry
Alias
Environment
FileSystem
Function
Variable
Certificate
WSMan
```

PS Providers

Providers	Data Store
Alias	PS built-in and user-defined aliases
Certificate	Windows digital signature certificates
Environment	Windows environment scalers
FileSystem	Windows file system drives, folders and files
Function	PS Functions
Registry	Windows Registry
Scaler	PS Scalers

```
PS C:\Windows\system32> Get-PSProvider Registry

Name                 Capabilities                  Drives
----                 ------------                  ------
Registry             ShouldProcess, Transactions   {HKLM, HKCU}
```

PowerShell's Scripting Language

SYED AWASE

Scripting

• Like many command-line shells, PS contains a scripting language, which is useful to automate complex, multipart processes that may require different actions to be taken for different scenarios.

•Language keywords are around 24 commands loosely modeled on C# and lends a strong resemblance to other C-based languages such as PHP, C++ and Java.

• Most of Scripting is based on conditions to evaluate expressions.

• A script is a convenient way of packaging a handful of commands to execute a specific job.

•A function is a wrapper around a set of commands, created in such a way that multiple functions can be listed within a single script file

•Scripts generally run with their own scope.

Simple Scripting Example (Looping)

```
#Forloop
For($i=0; $i -lt 10; $i++){
    Write-Host "pink money is the new black money:" $i
}
#Do-While Loop
$j=0
Do{
    $j
    $j++
    Write-Host "Panama Files" $j

}While($j -lt 15)
```

```
#Do-Until loop
$k=0
Do{
$k
$k++
Write-Host "Tweets Hacked-Vijaya Mallya"$k
} Until($k -eq 10)

#while-loop
While($m -lt 9){
    $m
    $m++
    Write-Host "Every day we stand in queue for money and we get"$m
}
```

```
#ForEach
$aks_services = Get-Service -name S*
ForEach($aks_service in $aks_services){
    $aks_service | select Name, DisplayName, Status
}
```

Simple Scripting Example(Conditions)

```
#ifelseif
If ($mood.Good) {
Get-Service
} else {
Restart-Computer -computername localhost
}
```

```
$procs = Get-Process | select Name, Handles, WS
foreach ($proc in $procs){
if (($proc.Handles -gt 200) -and ($proc.WS -gt 50000000)) {
$proc
    }
}
```

```
#switch
Switch ($printer.status) {
0 {$status = "OK"}
1 {$status = "Out of paper"}
2 {$status = "Out of Ink"}
3 {$status = "Input tray jammed"}
4 {$status = "Output tray jammed"}
5 {$status = "Cover open"}
6 {$status = "Printer Offline"}
7 {$status = "Printer on Fire!!"}
Default {$status = "Unknown"}
}
```

PowerShell Workflow

- Built on top of Windows Workflow foundation
- Uses Window WF to execute the scripts
- Two main goals with Workflow
  - Allow for long running tasks
  - Robust, interruptable

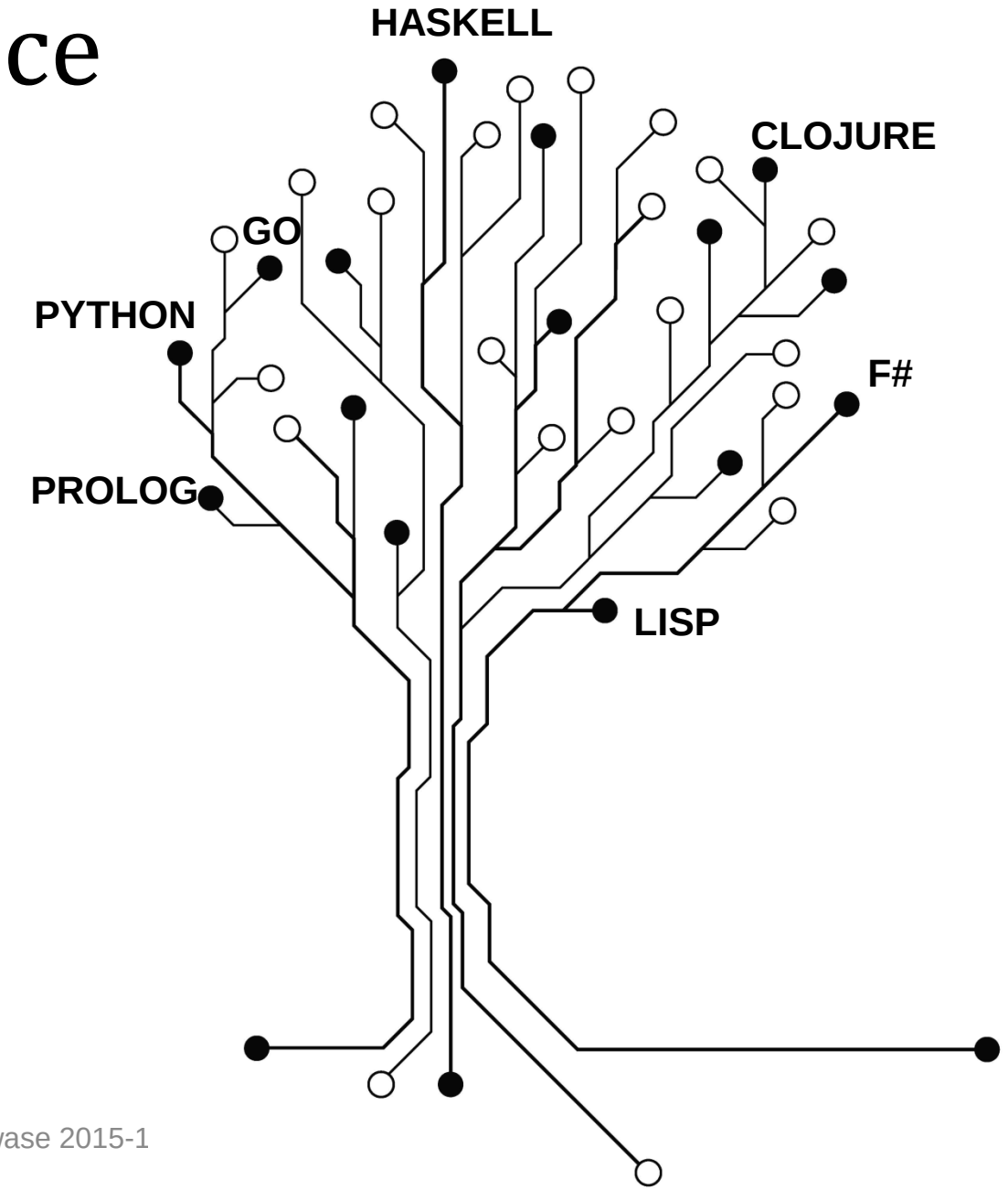
- Workflows use Stateless Execution
  - Each line of code in a workflow is considered an activity
  - Each activity is considered a standalone unit, i.e. when executed it is as if it is being run for the first time
  - This is necessary so a workflow can be suspended and resumed
  - Exception is with non-object variables, their values will be persisted.

Empowering You

TPRI-SYCLIQ PROGRAMS OVERVIEW

Artificial Intelligence

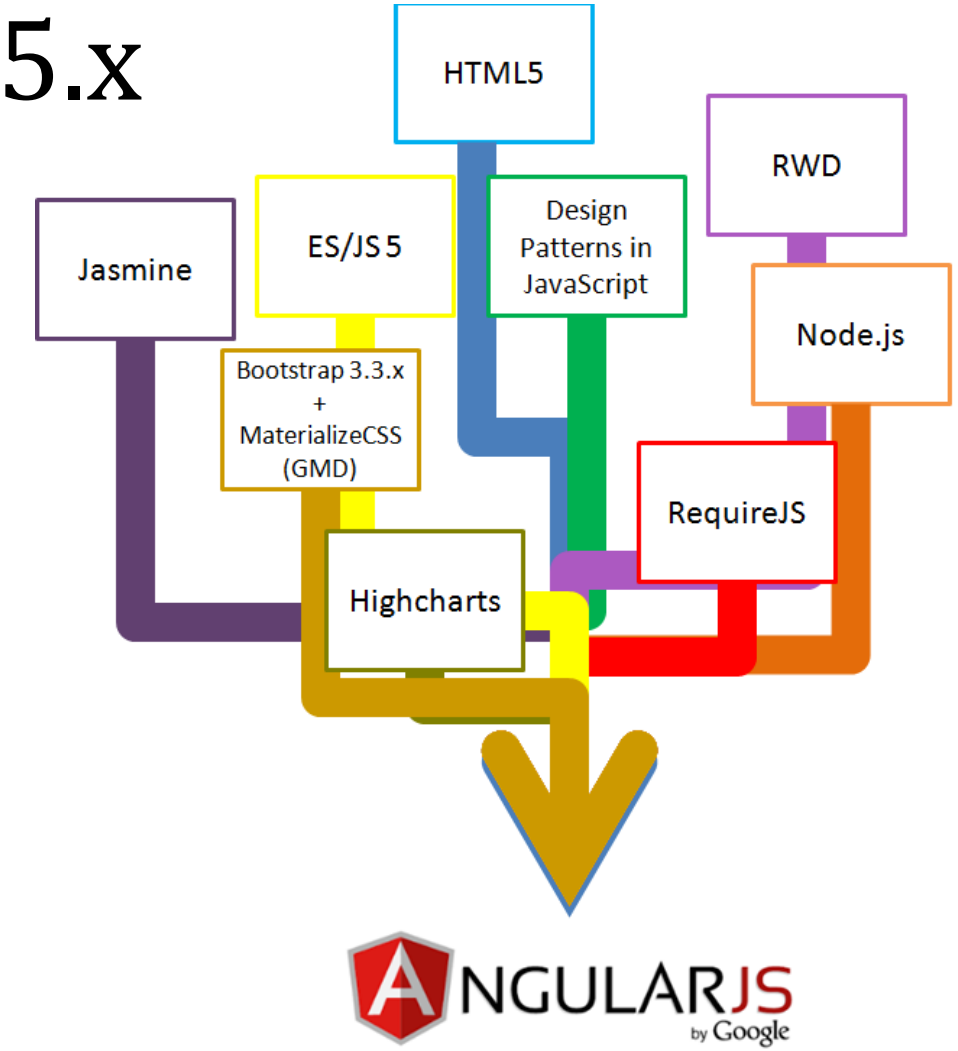
- We also train on AI Stack
- Reach out to us sak@sycliq.com or sak@territorialprescience.com
- www.territorialprescience.com
- www.sycliq.com
- +91.9035433124



Angular 2.x/Angular JS 1.5.x

Dr. Syed Awase 2016 Session Feedbacks: http://bit.ly/2hhNg58

Reach out to sak@territorialprescience.com/+91.9035433124



Dr. Syed Awase also offers Machine learning Stack, R Statistical Stack, .NET Stack, Java Stack, RaspberryPi Stack. Get the pulse of performance from here http://bit.ly/2hhNg58

NOW OFFERING!

Code Focused Training

JAVASCRIPT FRAMEWORKS
CODE DRIVEN CAPACITY BUILDING PROGRAM

Dr. Syed Awase 2016 Session Feedbacks: http://bit.ly/2hhNg58

Reach out to sak@territorialprescience.com/+91.9035433124

Dr. Syed Awase also offers Machine learning Stack, R Statistical Stack, .NET Stack, Java Stack, RaspberryPi Stack. Get the pulse of performance from here http://bit.ly/2hhNg58

TERRITORIAL™ PRESCIENCE
provoke thought, invoke action

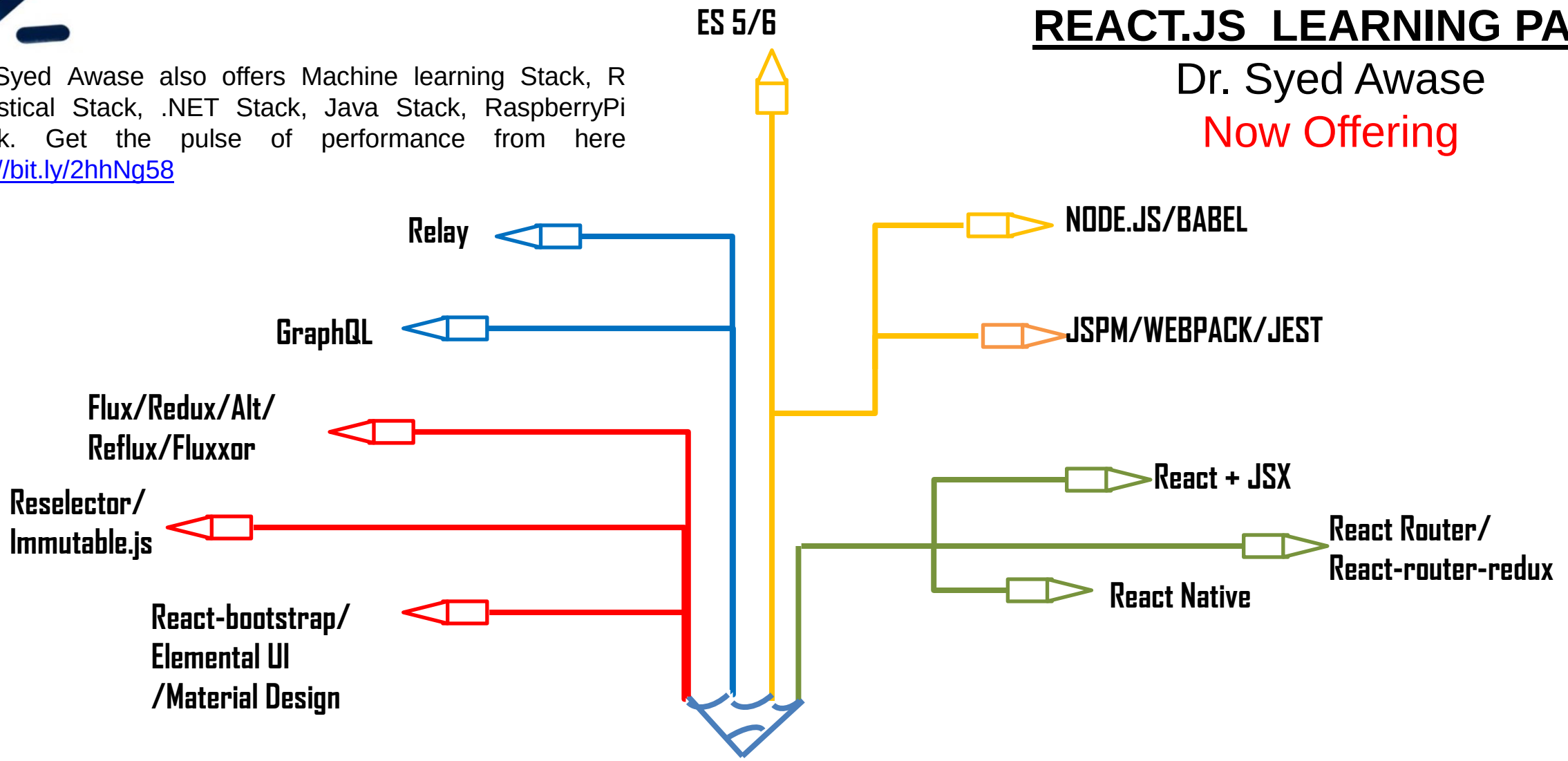
Code Focused Training

REACT.JS LEARNING PATH

Dr. Syed Awase

Now Offering

Dr. Syed Awase also offers Machine learning Stack, R Statistical Stack, .NET Stack, Java Stack, RaspberryPi Stack. Get the pulse of performance from here http://bit.ly/2hhNg58



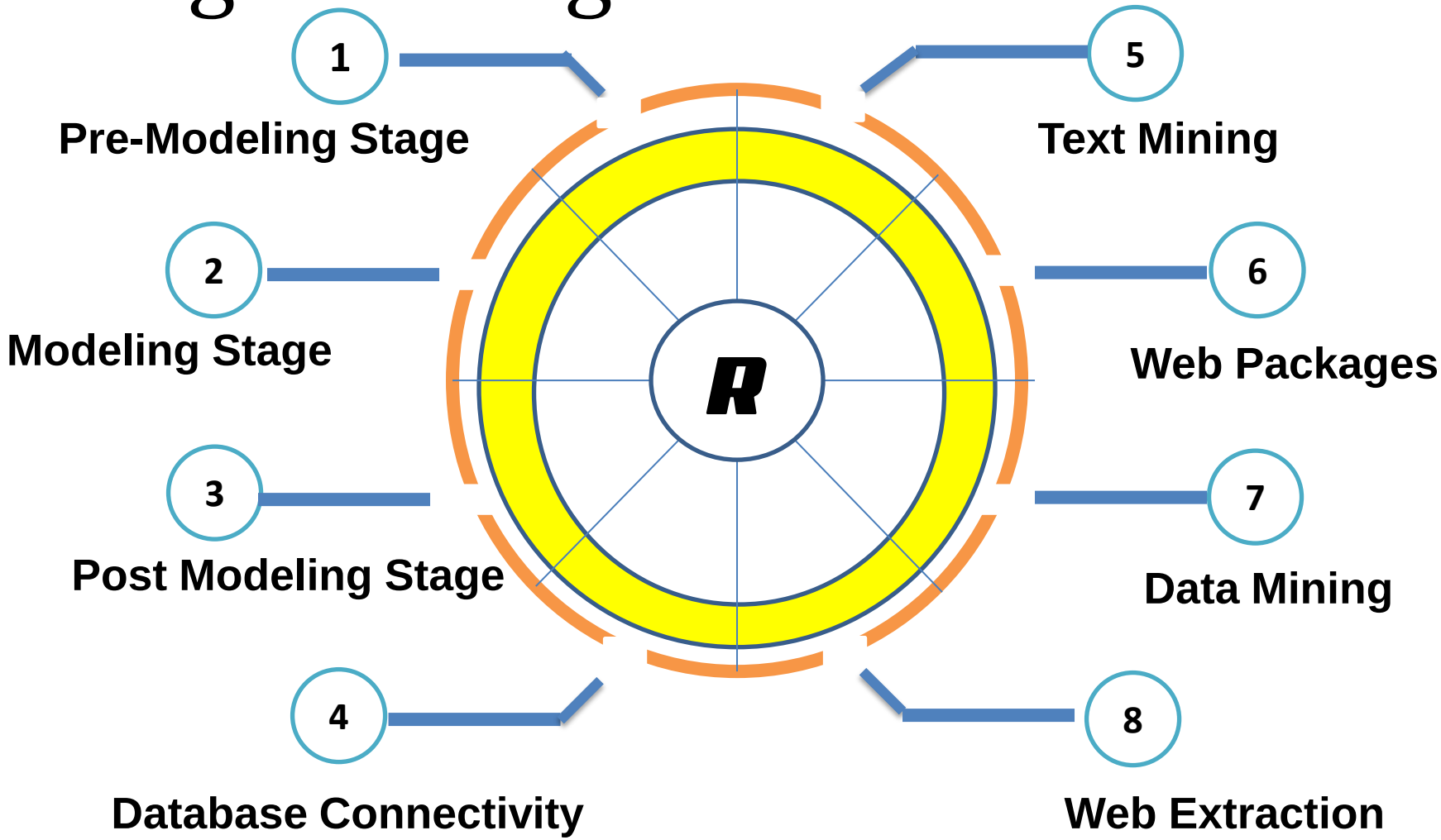
TERRITORIAL™ PRESCIENCE
provoke thought, invoke action

Reach out to us @ sak@territorialprescience.com / +91.9035433124

R-Statistical Programming

Dr. Syed Awase 2016 Session Feedbacks: http://bit.ly/2hhNg58

Reach out to
sak@territorialprescience.com/
+91.9035433124



Dr. Syed Awase also offers Machine learning Stack, R Statistical Stack,
.NET Stack, Java Stack, RaspberryPi Stack. Get the pulse of performance from here http://bit.ly/2hhNg58

For code driven trainings for Technology Firms reach out to us +91-9035433124
We are hardcore Technologists/Architects/Programmers
trainings are offered by Dr. SYED Awase

Code Focused Training

TERRITORIAL PRESCIENCE
provoke thought, invoke action

Thank You

We also provide Code Driven Open House Trainings : sak@territorialprescience.com or sak@sycliq.com

Java Technologies

- Core Java
- Hibernate
- Spring Framework
- Play Framework
- Hadoop
- Groovy & Grails

Microsoft Technologies

- C# Core
- Entity Framework
- MVC 5/6
- Web Api
- OWIN/KATANA
- WCF
- WPF

Python

- Python
- Django
- Flask
- Numpy
- Scipy
- Machine Learning

Data Science

- R Statistical Programming
- Julia

SQL and NoSQL

- Oracle
- PostgreSQL
- MSSQL
- MongoDB
- Neo4j
- Redis
- Firebase
- Apache Cassandra

Client-Side Frameworks

- Angular JS 1.5.x
- Angular 2.4.x
- React JS
- KnockOut JS
- VueJS
- Backbone JS
- EMBER JS
- Hapi JS
- METEORJS
- MEANJS
- Coffeescript
- Dart

Others

- LISP
- CLOJURE
- RUST
- GO
- RaspberryPI
- Coming Soon
  - PHP
  - Robotic OS