A BEGINNER'S GUIDE TO USING MICROSOFT DESIRED STATE CONFIGURATION

DSC LABS

OBJECTIVES

- Learn DSC by using it to deploy and manage lab components
- Learn to create Self-Signed certificates for Windows Server 2012 R2 and Windows Server 2016
- Learn to create CA-signed certificates with your own cert server that we build together
- Learn to create an https pull server and start using it

SYSTEM / SOFTWARE REQUIREMENTS

- VMware ESXi host
- VMware vCenter Server
- Template created for desired Windows version(s)
- Linux ISO or templates for Ubuntu 16.04 and CentOS 7, to optionally test **dsc** with **omi**.
- PowerShell 5.1
- VMware PowerCLI
- Compute and storage for virtual machines

Note: Skip these requirements if you will provide your own guests to test with.

EXPERIENCE

- Basic PowerShell skills required
- Also, recommend going through the following PluralSight training courses first:

- Windows PowerShell Desired State Configuration Fundamentals
- Advanced Windows PowerShell Desired State Configuration
- Practical Desired State Configuration (DSC)

PRIOR ART

- DSC_LABS is based on the common DSC examples provided with official and community resources
- Also, DSC_LABS is very much based on the course mentioned earlier “Practical Desired State Configuration (DSC).
- We have updated the examples for certificate handling to reflect the modern versions of the PSPKI and PKI modules.

LAB SETUP

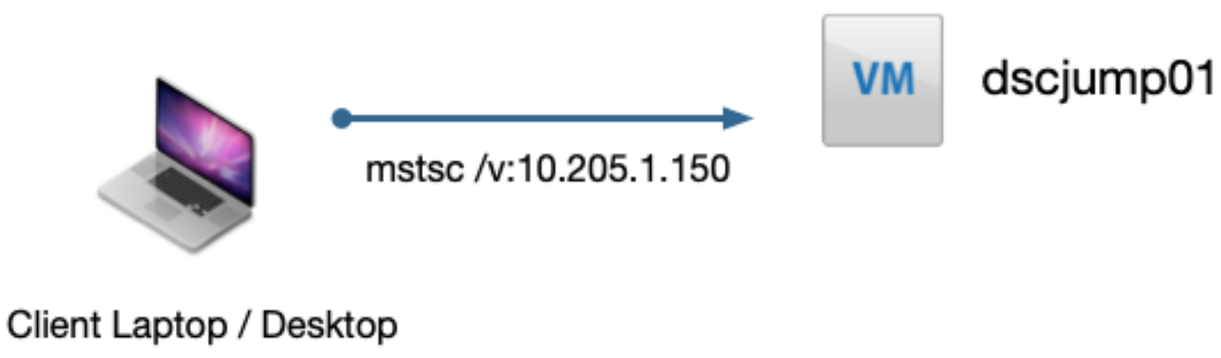
JUMP SERVER

Use PowerCLI on your client to login to vCenter

Deploy a jump server for use with DSC

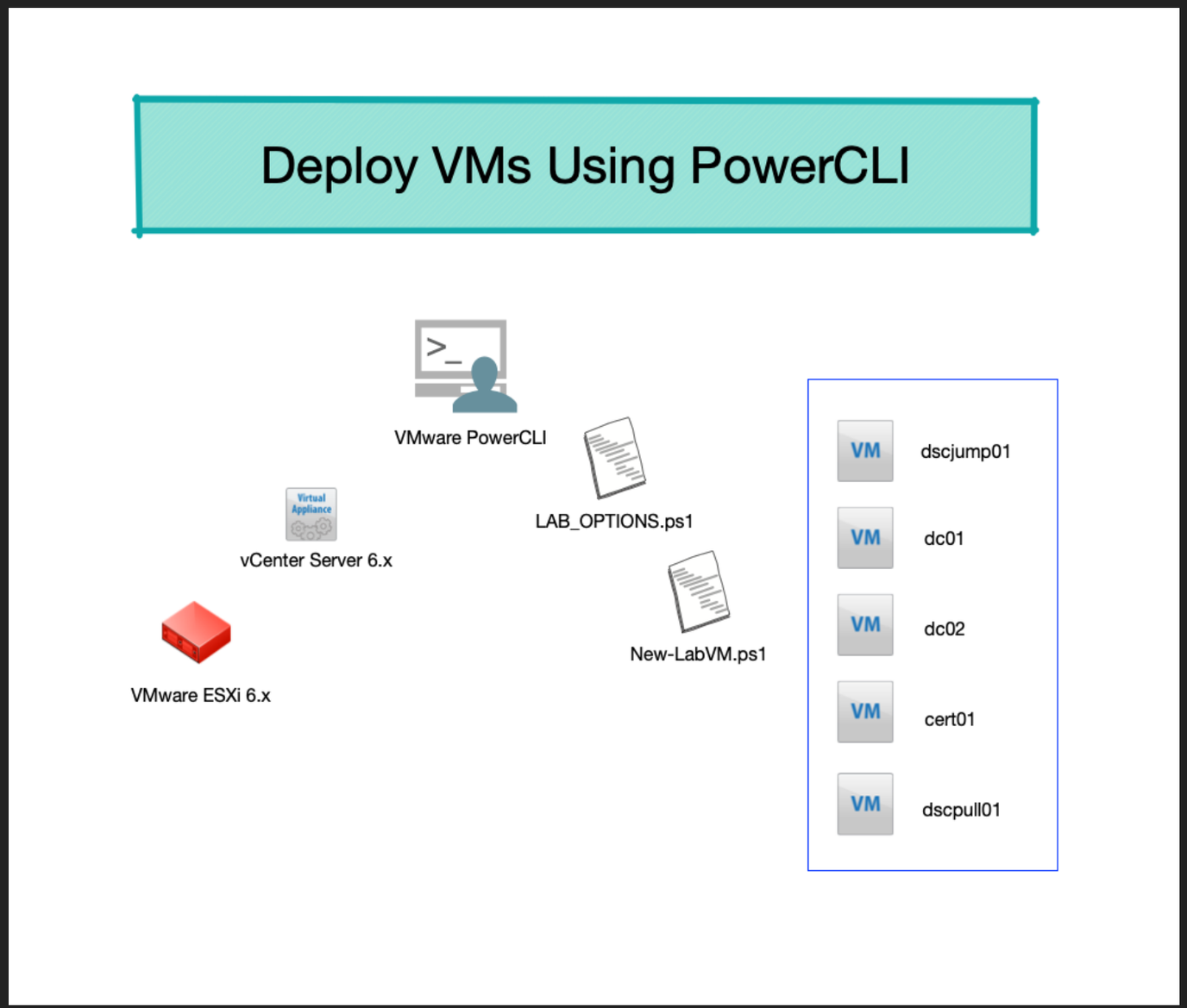
Optionally, use the **New-LabVM** script to deploy virtual machines

Finally, RDP to jump server by IP Address



New-LabVM

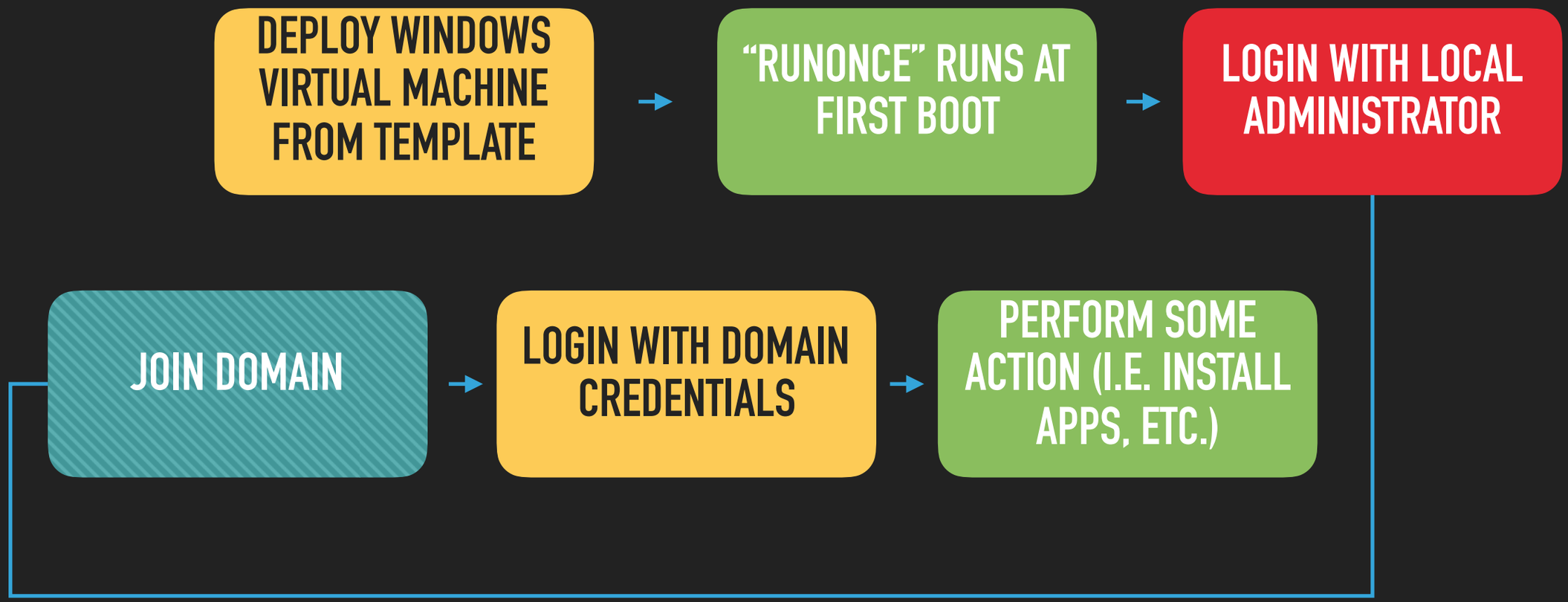
Optionally, use the **New-LabVM** function to deploy one or more virtual machines.
There is no DSC at this point yet, just pure PowerCLI to deploy the components.



DEPLOY YOUR OWN

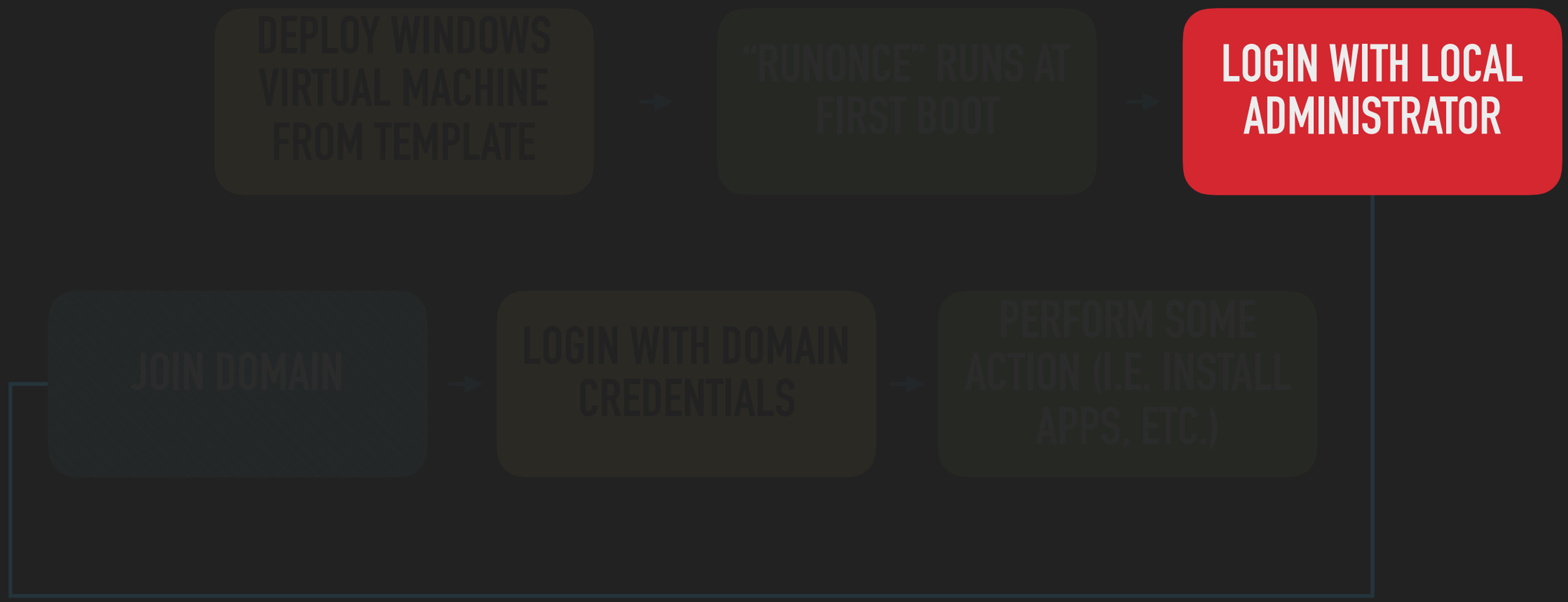
- If you will not use New-LabVM, then you can deploy your own VMs as desired.
- This is fine since we are not using DSC yet.
- Once the desired virtual machines are deployed, we will then configure each node together with DSC.

Typical deployment model



FOCUS ON IMPORTANT PARTS, LIKE LOGIN.

After template deploy - Login with the RID500, true Administrator.



LOGGING IN AS LOCAL ADMINISTRATOR

- Where possible, always provide the machine name or IP Address and then the account.
- Do not use .**\Administrator**
- Instead, use “**10.205.1.150\Administrator**”
- or, “**dscjump01\Administrator**”

BASTION HOSTS

- For our purposes, a "**Bastion Host**" is anything that is not joined to the domain
- When we work with a freshly deployed guest from a Windows template, that is a bastion host
- Once we join the domain, then we treat it differently of course
- This becomes important when dealing with self-signed vs. domain certificates
- The examples walk through the configuration of a **Bastion 2012 R2** and **Bastion 2016 server.**
- Eventually, we deploy our own CA in lab.local and use a CA-signed certificate instead of the self-signed bastion certs.

CERTIFICATES AND HOW TO GET THEM

- One could purchase a real certificate but that is not required
- In the demos, we build our own, using a CA that we create
- For initial build, we will use a self-signed certificate that we manually generate on each node
- Once a guest is domain joined, we can use a certificate from the lab.local domain that we generate

DEPLOYMENT OVERVIEW

For initial build, we use Self-Signed Certs that we generate

SELF-SIGNED CERTIFICATES

dc01 dc02 cert01 dscpull01

Later, we use the pull server to manage all nodes, using a cert from lab.local

CA-SIGNED CERTIFICATES

dc01 dc02 cert01 dscpull01

s1 s2 ..n

CREATE A CERTIFICATE ON WINDOWS SERVER 2016

- The official Microsoft “PKI” module is native on Windows Server 2016 and later
- This gives us the “**New-SelfSignedCertificate**” function without any extra work

Windows Server 2016, Bastion Example

```
New-SelfSignedCertificate `
-Type DocumentEncryptionCertLegacyCsp `
-DnsName 'DscBastionCert' `
-HashAlgorithm SHA256
```

CREATE A CERTIFICATE ON WINDOWS SERVER 2012 R2

- Creating certificates for Windows Server **2012 R2** or older is more **difficult**
- We need to add a **community** module from the **PSGallery** called **PSPKI**.
- PSPKI gives us the "**New-SelfSignedCertificateEx.ps1**"
- Note the "Ex" at the end of the name, compared to the official **PKI** function **New-SelfSignedCertificate.ps1**.
- If not using the **PSGallery** on remote nodes, we need to use **Copy-Item** or similar to get the bits to the guest.

Windows Server 2012 R2, Bastion Example

```powershell
# Install the PSPKI module using gallery
Find-Module -Name PSPKI | Install-Module

# Variables for new certificate to create
[string]$Subject = ('CN={0}' -f $session.ComputerName)
[string]$StoreLocation = 'LocalMachine'
[string]$KeyUsage  = 'KeyEncipherment'
[string]$EnhancedKeyUsage =  '1.3.6.1.4.1.311.80.1'
[string]$FriendlyName = 'DscBastionCert'
[datetime]$NotBefore = [datetime]::now.AddDays(-1)
[datetime]$NotAfter = [datetime]::now.AddDays(1)  #or AddYears(1)
[string]$SignatureAlgorithm = 'SHA256'


# Create certificate
  New-SelfSignedCertificateEx `
  -Subject $Subject `
  -StoreLocation $StoreLocation `
  -KeyUsage $KeyUsage `
  -EnhancedKeyUsage $EnhancedKeyUsage `
  -FriendlyName $FriendlyName `
  -NotBefore $NotBefore `
  -NotAfter $NotAfter `
  -SignatureAlgorithm $SignatureAlgorithm `
  -Verbose:$true
```

ABOUT LEGACY PSPKI VERSIONS

Needed for Windows Server 2012 R2 (or older) to create self-signed certificates.

The PSPKI module contains the function “New-SelfSignedCertificateEx”.

Formerly, this was a stand-alone script and not part of a module.

The author supports the latest 2 versions of the function.

Only the module versions are supported

By support, this is community support.

Note: If your test VMs are domain joined already, then you can probably skip this.

HANDLING TIME

PROBLEM

time.windows.com

UDP 123 blocked

- New Windows Server
- Guest cannot get time
- Results in guest using UTC time
- Breaks kerberos

SOLUTION

- Use an internal time source

Corporate NTP

- Can be Active directory (i.e. IP Address of PDC Emulator). This is technically SNTP.
- Can be a network switch, if supported and running NTP.
- May need to correct time with the network switch first, and then switch to NT5DS.

PROBLEM: W32TIME DISABLED

BACKGROUND: Fresh deploy of modern Windows Server will have the Windows Time Service set to '**Disabled**' by default.

EXAMPLE SCRIPT

```powershell
## Get w32Time service info
$timeSvc = Get-Service -Name W32Time
$timeSvcStartType = $timeSvc | Select-Object -ExpandProperty StartType

## Sets the StartType for the w32time service
If($timeSvcStartType -match '^Disabled'){
    try{
      $null = $timeSvc | Set-Service -StartupType 'Manual' -Confirm:$false -ErrorAction Stop
    }
    catch{
      Write-Warning -Message 'Problem setting w32Time service to Manual start!'
      Write-Error -Message $Error[0].exception.Message
    }
}
```

*Note: The above snippet is available in the **New-WinFwRule.ps1** script, included with **DSC_LABS**.*

LAB BASICS

- Windows Templates (i.e. Server 2012 R2 and Server 2016)
- Linux ISOs (i.e. CentOS 7 and Ubuntu 16.04)
- local login for new machines (i.e. Administrator or root, or other created account).
- Reachable time source (optional, recommended)

GET STARTED

Extract the DSC_LABS.zip to C:\

Navigate to “C:\DSC_LABS\Docs”

Open up “DEMO 1“ to get started.

Each demo runs in order to create all components

Note: We’ll need both the Microsoft ISE and Visual Studio Code to follow along.

LABS / DEMOS

QUESTIONS

End