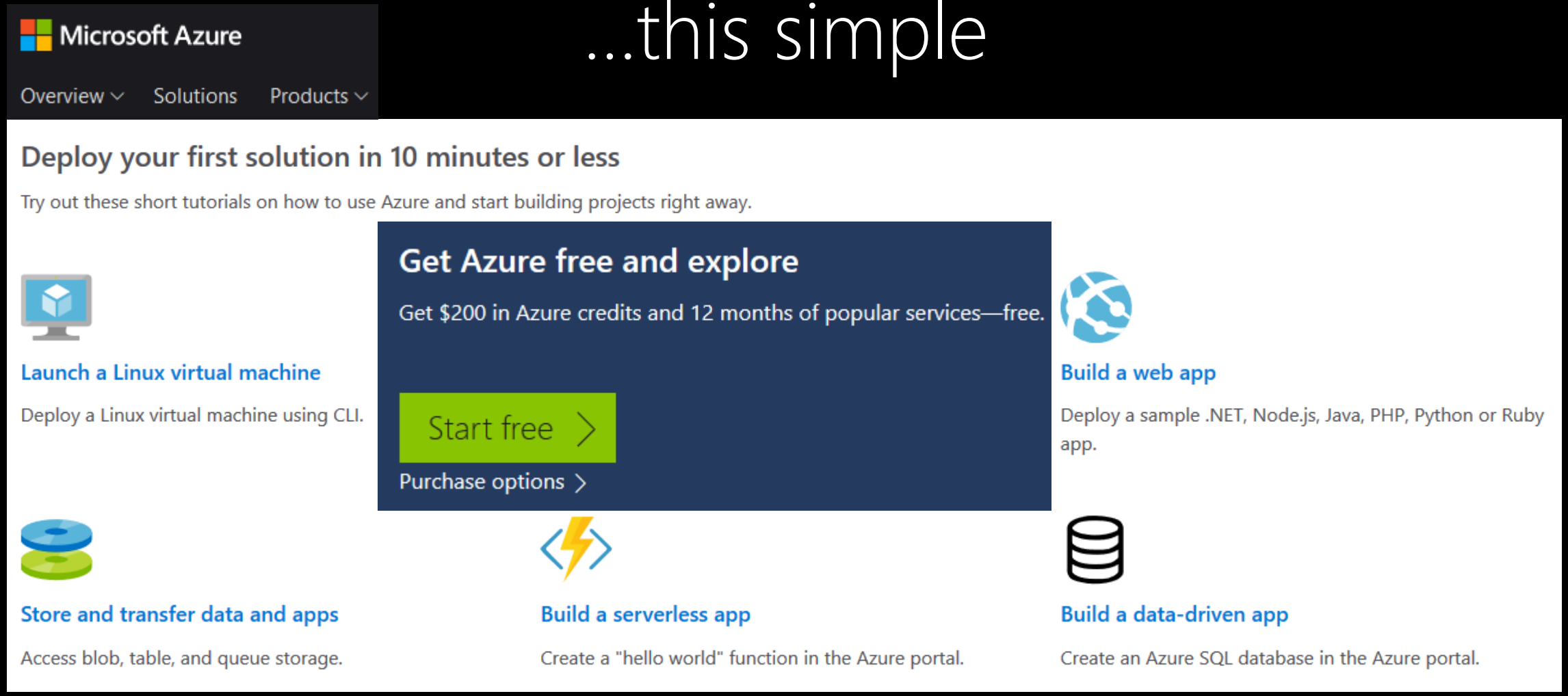
Microsoft
tech·days
Kistamässan Stockholm
24-25 oktober 2018

Secure infrastructure with Terraform, Azure DSC and Ansible

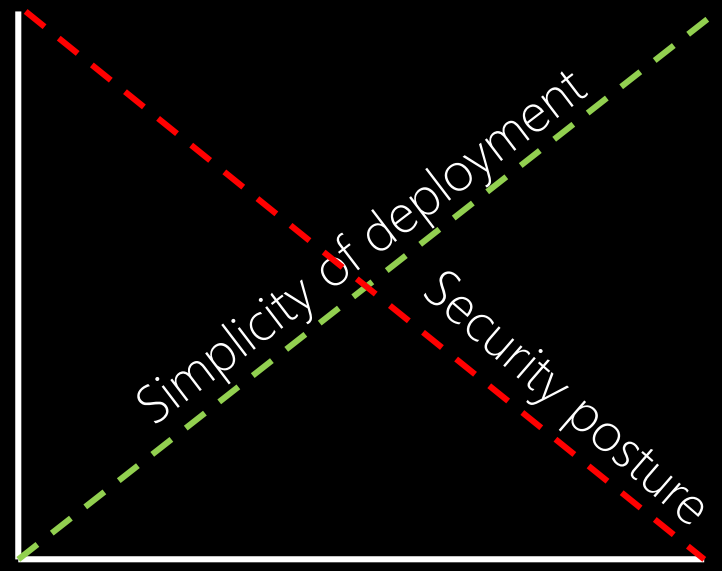
Igor Andriushchenko

Building infrastructure has never been…

…this simple

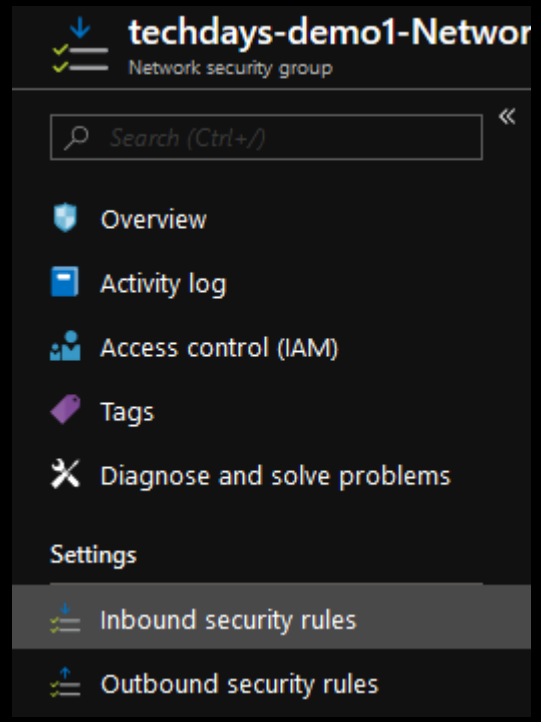


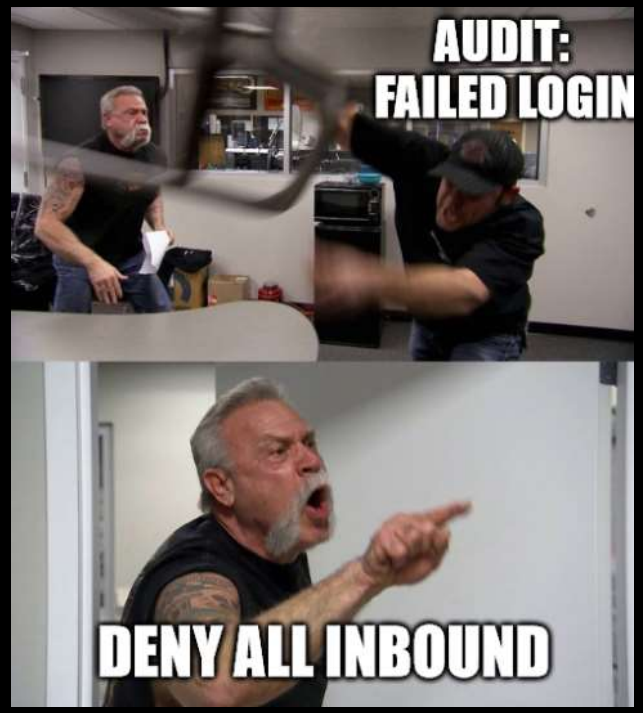
Security needs to catch up



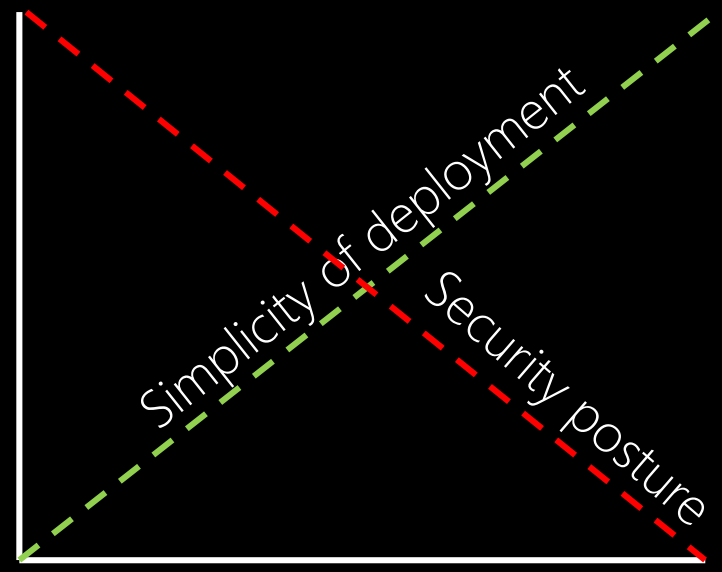
Security in 2010

Security in 2018





Security needs to catch up



…automation is the last hope

About me

Wrote my first code 20 years ago

Worked with Azure since 2012

Survived 3 years in Finland

NOKIA

GE Healthcare

Security Lead at Snow Software

@doshch cloudappsec.net

What this talk is NOT about…

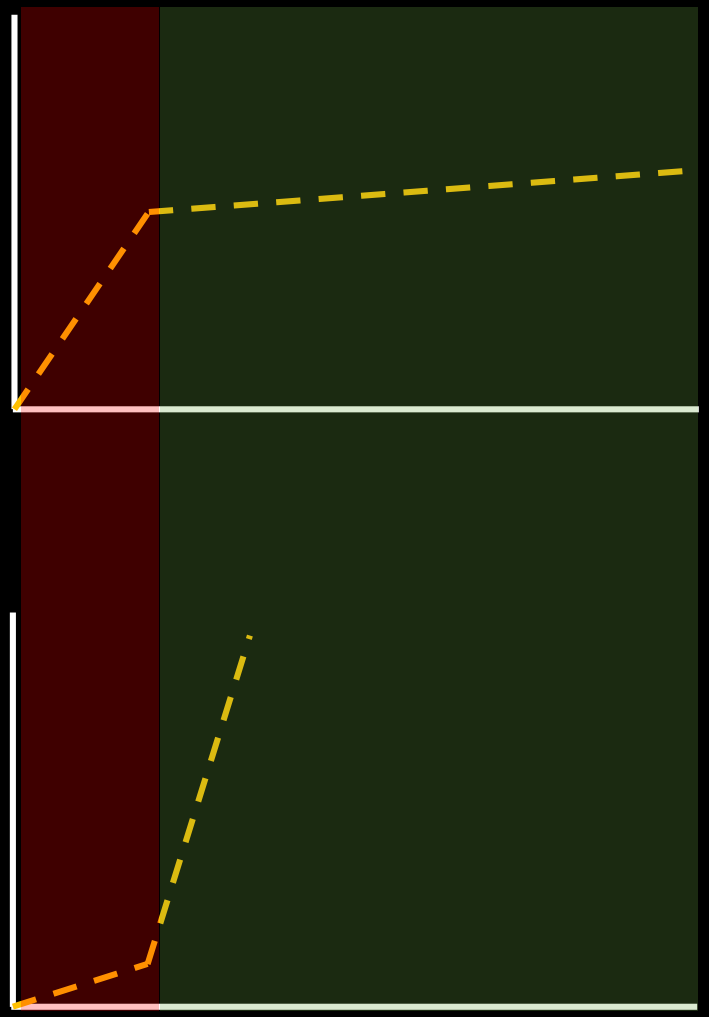
This talk is about:

A tool rather than THE tool

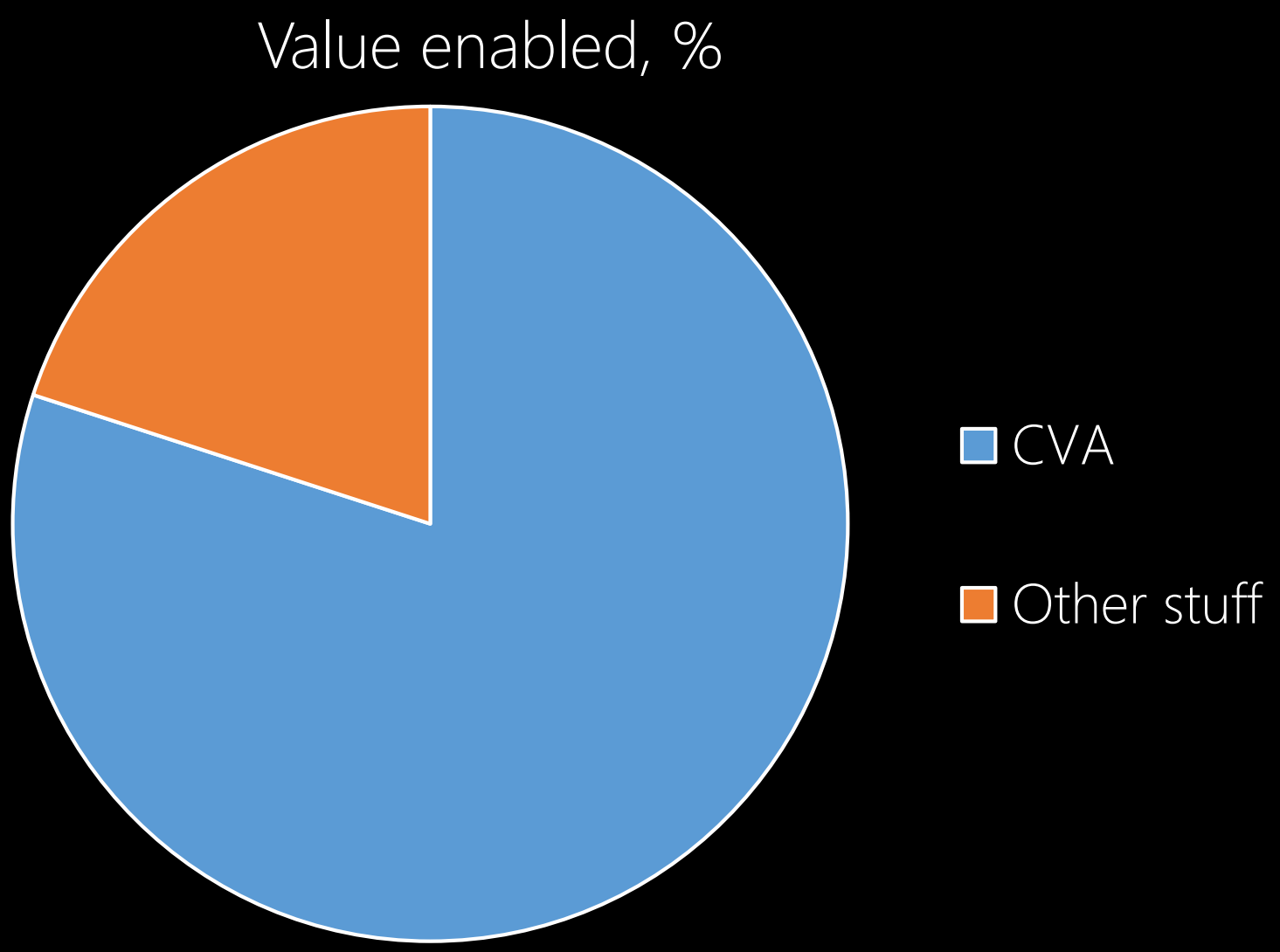
Modular rather than monolithic

Showing rather than telling

Learning curve with automated tools



Security value added with tools



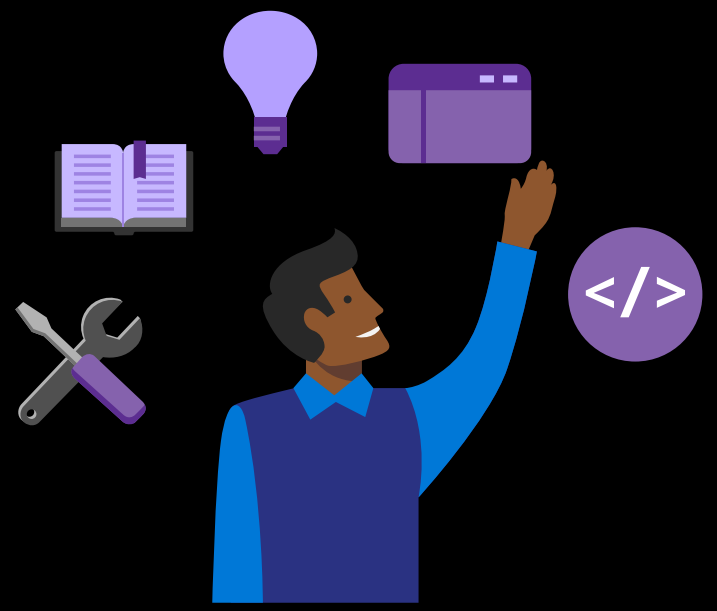
Focus on:

Control

Visibility

Automation

Meet John



He is tasked with building secure infra on scale

John's current infrastructure

- Manually built
- Start from 0 in case of disaster
- State is unknown
- Inconsistent
- Changes are slow
- Doesn't scale

Secure Infrastructure

Codified

Easily redeployable

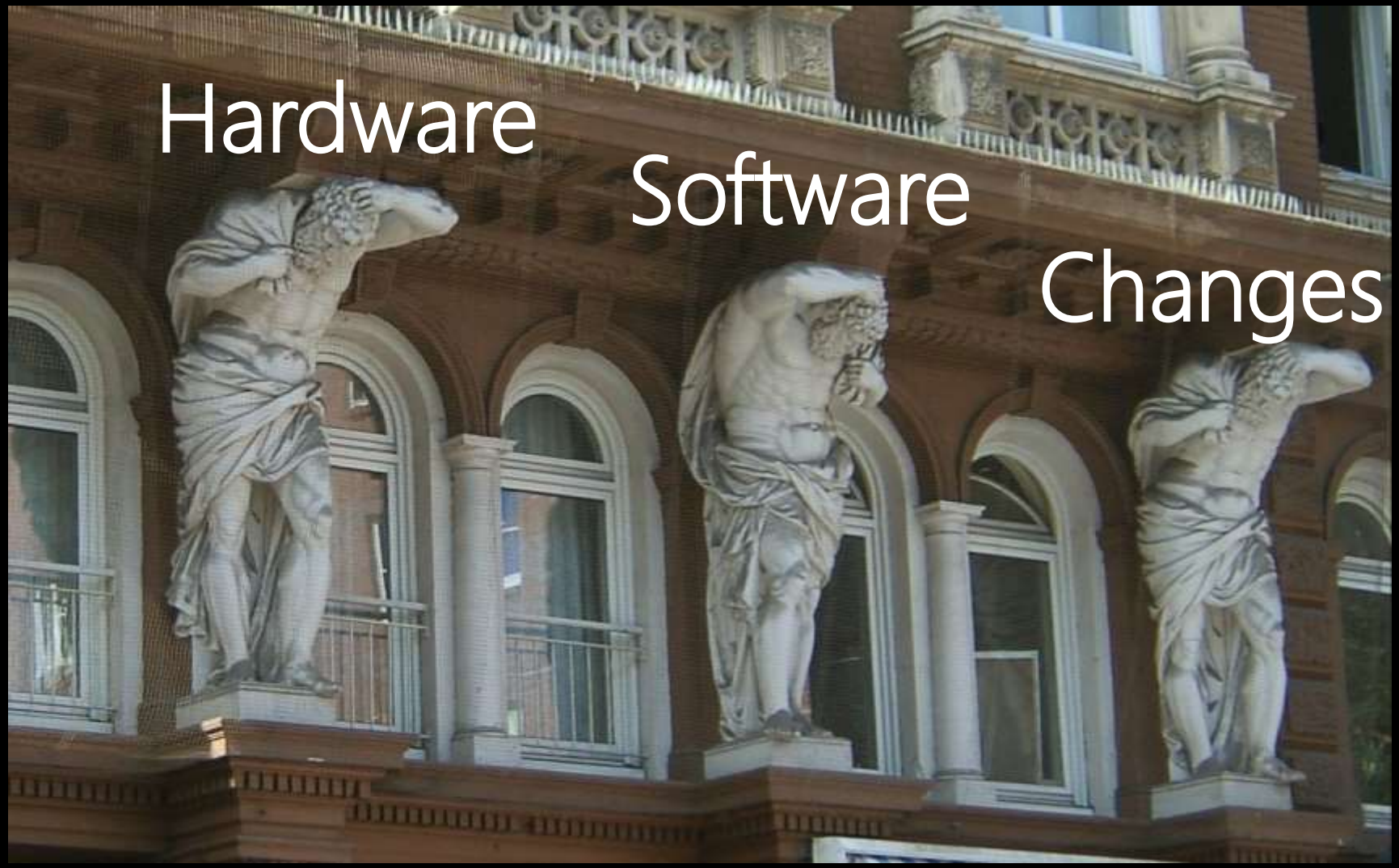
Insightful

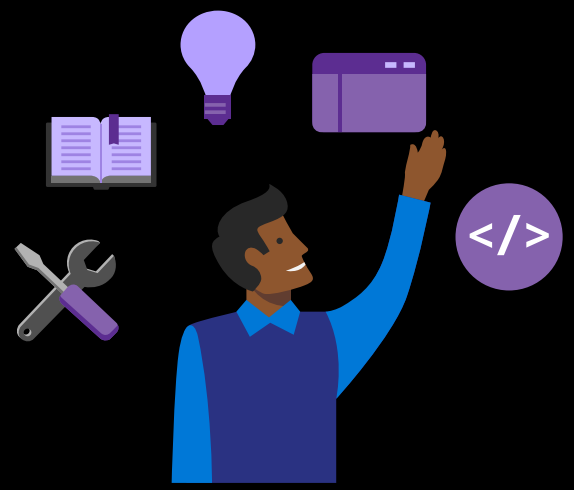
Unified

Quick and wide changes

Unlimited scaling

3 pillars of infrastructure





Changes

Changes

Changes

Changes

Changes

Changes

State

State

Hardware

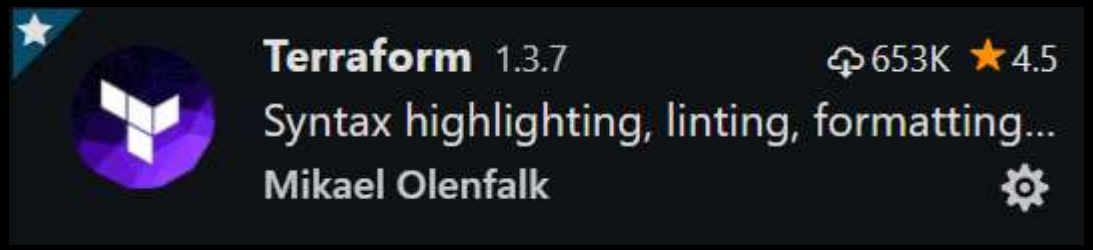
Terraform – cloud as code

- Infrastructure as Code
- 90+ providers for actual technologies: Azure, AWS, GCP, VMware, k8s, Docker
- Computes accurate resource representation from a human-readable template
- Source-control and collaboration friendly
- Built for automation

Set up work environment

VS Code

Terraform extension



terraform.exe

plan

apply

.tf

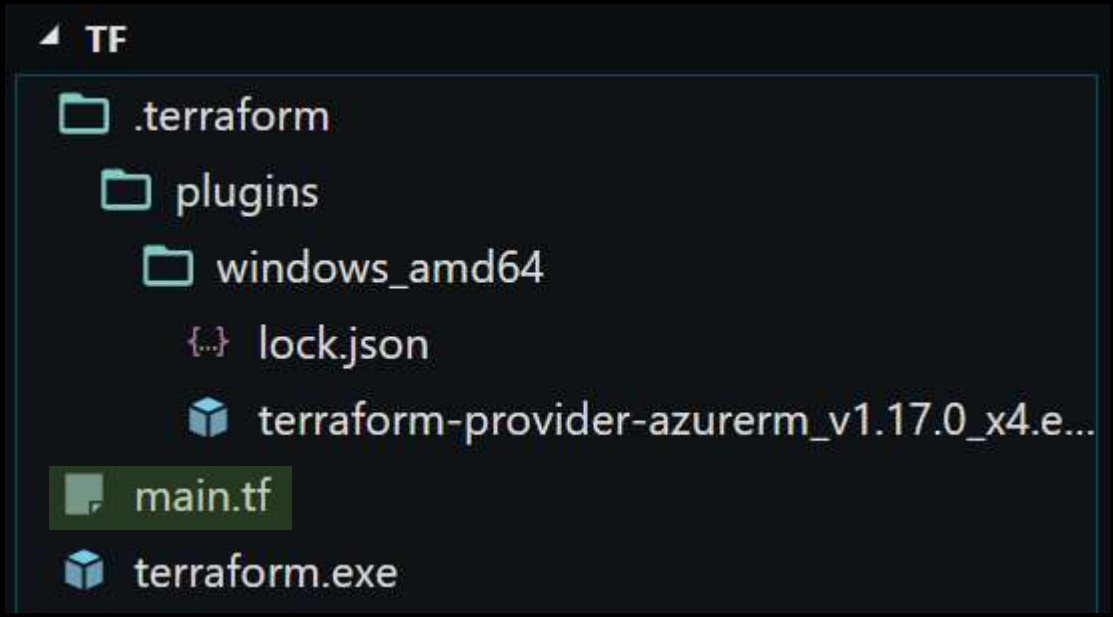
.tfstate

Initializing Terraform

Standalone binary with required plugins

> terraform.exe init

```
- Downloading plugin for provider "azurerm" (1.17.0)...
```



Or using Azure Cloud Shell

```
PS Azure:\> terraform init
Terraform initialized in an empty directory!

The directory has no Terraform configuration files. You may begin working
with Terraform immediately by creating Terraform configuration files.
Azure:/
PS Azure:\>
```

Write in integrated editor in browser

main.tf

```
resource "azurerm_resource_group" "demorg" {
        name = "techdays-${var.targetName}-rg"
        location = "${var.location}"

        tags {
            environment = "techdays-${var.targetName}"
        }
}
```

Plan file: .tf or .tf.json

Resources + variables + outputs

```
7 references
resource "azurerm_resource_group" "demorg" {
    name = "techdays-${var.targetName}-rg"
    location = "${var.location}"

    tags {
        environment = "techdays-${var.targetName}"
    }
}
```

Resource type

Resource internal name

Resource parameters

Variables

```
variable "targetName" {
    default = "demo1"
}
variable "location" {
    default = "northeurope"
}
resource "azurerm_resource_group" "demorg" {
        name = "techdays-${var.targetName}-rg"
        location = "${var.location}"

        tags {
            environment = "techdays-${var.targetName}"
        }
}
```

Variable name

Variable value

Variable interpolation

Outputs

Resource parameters interpolation

```
resource "azurerm_public_ip" "demopublicip" {
    name                          = "techdays-${var.targetName}-publicIP"
    location                      = "${var.location}"
    resource_group_name           = "${azurerm_resource_group.demorg.name}"
    public_ip_address_allocation = "static"

    tags {
        environment = "techdays-${var.targetName}"
    }
}
```

```
output "IpAddress" {
  value = "${azurerm_public_ip.demopublicip.ip_address}"
}
```

Dependencies

```
depends_on                  = ["azurerm_virtual_machine.demovm"]
```

> terraform.exe init

> az login

> terraform.exe plan

```
+ azurerm_resource_group.demorg
    id:                                <computed>
    location:                          "northeurope"
    name:                              "techdays-demo1-rg"
    tags.%:                            "1"
    tags.environment:                  "techdays-demo1"
```

```
Plan: 8 to add, 0 to change, 0 to destroy.
```

```
{
    "version": 3,
    "terraform_version": "0.11.7",
    "serial": 5,
    "lineage": "fb5c0f30-2112-ac8b-2854-0548786a6275",
    "modules": [
        {
            "path": [
"primary": {
    "id":
    "/subscriptions/7858a8d4-7ded-4317-ad1e-a19ccc727e78/resourceGroups/techdays
    mo1-rg/providers/Microsoft.Network/networkInterfaces/techdays-demo1-nic",
    "attributes": {
        "applied_dns_servers.#": "0",
        "dns_servers.#": "0",
        "enable_accelerated_networking": "false",
        "enable_ip_forwarding": "false",
        "id":
        "/subscriptions/7858a8d4-7ded-4317-ad1e-a19ccc727e78/resourceGroups/tech
        s-demo1-rg/providers/Microsoft.
```

Apply changes from calculated .tfstate

> terraform.exe apply

```
An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create
```

```
azurerm_virtual_machine.demovm: Creating...
  availability_set_id:                                    "" => "<computed>"
  delete_data_disks_on_termination:                       "" => "false"
  delete_os_disk_on_termination:                          "" => "false"
  identity.#:                                             "" => "<computed>"
```

Failed deployments can be resumed

Corrected TF template == updated .tfstate

> terraform.exe apply

```
azurerm_resource_group.demorg: Refreshing state... (ID: /sub
azurerm_network_security_group.demonsg: Refreshing state...
up)
```

```
Plan: 2 to add, 0 to change, 0 to destroy.
```

If a wrong resource was deployed, it needs to be removed first

When removing resources, TF looks into .tfstate

> terraform.exe destroy

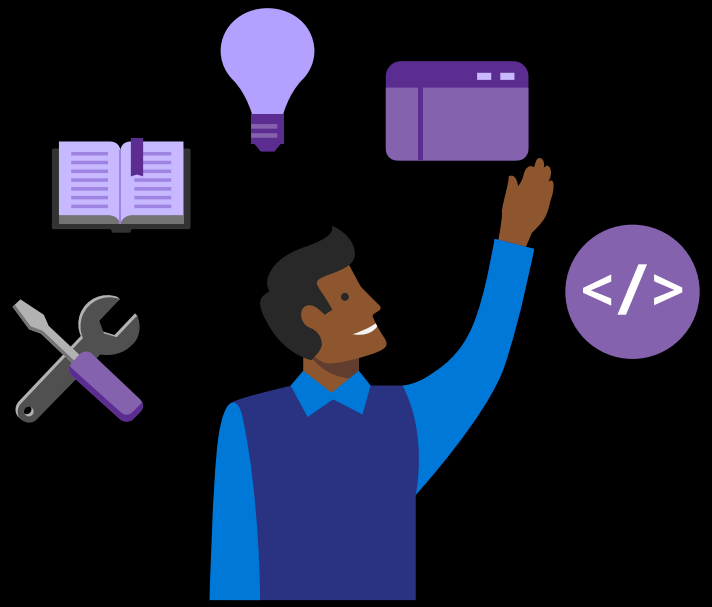
```
Terraform will perform the following actions:

  - azurerm_resource_group.demorg


Plan: 0 to add, 0 to change, 1 to destroy.
```

Helps clean up failed deployments

Considerations

- Never ever ever lose .tfstate
- Check-in .tf and .tfstate files
- Both .tf and .tfstate may contain secrets!
- Variables can be kept in a file, sourced from KeyVault or specified in command line

Demo: Deploy simple VM



John wants to deploy all VMs programmatically

DSC – controlling the state

PowerShell Desired State Configuration… as a service:

PSDSCaaS

What is Azure DSC

- DSC pull server compiles and stores configurations
- Part of Azure Automation
- Requires VM extension installed
- Pay for node communication: ~$0.25 / node / month

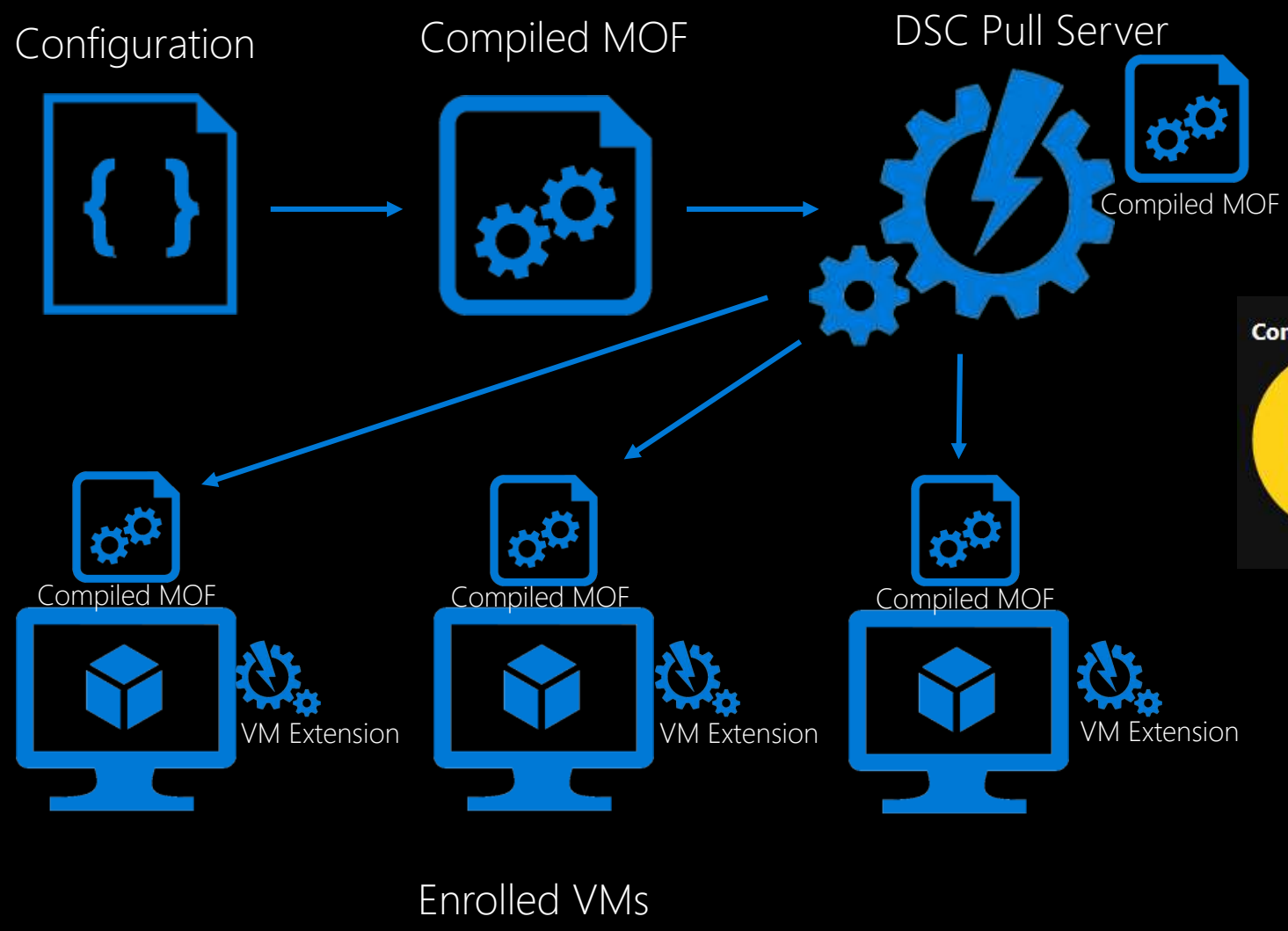
Azure DSC prerequisites

Windows Server 2008 R2

PS 4.0

AzureRM-deployed VM
(no classic)

How Azure DSC works



- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems

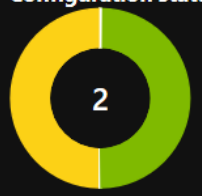
Configuration Management

- Inventory
- Change tracking
- State configuration (DSC)

Add | Refresh | Reset filters | Log search

Nodes | Configurations | Compiled configurations | Gallery

Configuration status



Failed 0	Pending 1
Not compliant 0	In progress 0
Unresponsive 0	Compliant 1

Nodes: Search Node names...

Status: 6 selected

Node configuration: All

NODE	STATUS	NODE CONFIGURATION	LAST SEEN
techdaysdemo1	Compliant	TechDaysDemo1.localhost	10/23/2018, 12:08 AM
techdaysdemo2	Pending	TechDaysDemoHardened.localhost	10/23/2018, 12:13 AM

```
"ConfigurationMode": "ApplyAndAutoCorrect",
"ConfigurationModeFrequencyMins": 15,
"RefreshFrequencyMins": 30,
"RebootNodeIfNeeded": true,
"ActionAfterReboot": "continueConfiguration",
"AllowModuleOverwrite": true
```

Configurations

```
Configuration TechDaysDemo1
{
    Import-DscResource -ModuleName PSDesiredStateConfiguration
    Import-DscResource -ModuleName xWindowsUpdate

    node "localhost"
    {
        xWindowsUpdateAgent ApplySecurityUpdates
        {
            IsSingleInstance = 'Yes'
            UpdateNow        = $true
            Source           = 'WindowsUpdate'
            Notifications    = 'ScheduledInstallation'
        }

        WindowsFeature WebServer
        {
            Name = 'Web-Server'
            Ensure = 'Present'
        }
    }
}
```

Import DSC modules

Target nodes

DSC resource

Check and apply all available Security updates

Install Web-Server Windows feature

Useful DSC modules

- NetworkingDsc
- ComputerManagementDsc
- PSDesiredStateConfiguration
- xPSDesiredStateConfiguration

x - experimental

More modules:

https://github.com/PowerShell

Install module to Automation account

Automate-7858a8d4-7
Automation Account

Search (Ctrl+/)

- Overview
- Activity log
- Access control (IAM)

...

Shared Resources

- Schedules
- Modules
- Modules gallery
- Python 2 packages
- Credentials

xWindowsUpdate
PowerShell Module

Import

Module with DSC Resources for Windows Update

Created by: PowerShellTeam
Tags: DesiredStateConfiguration DSC DSCResourceKit DSCResource PSModule
View Source Project

Version: 2.7.0
6,407,782 **downloads**
Last updated: 7/13/2017

Learn more

View in PowerShell Gallery
Documentation
Licensing information

Content

Search to filter items...

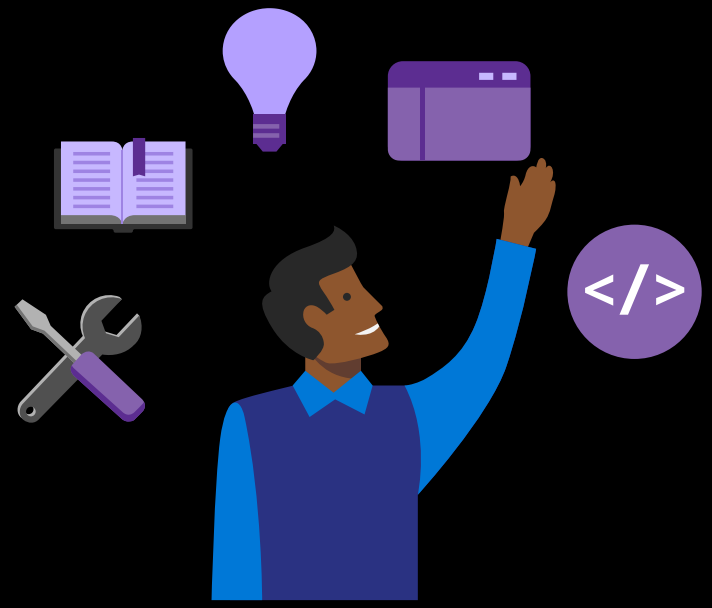
TYPE	NAME
DSC Resource	xHotfix
DSC Resource	xMicrosoftUpdate
DSC Resource	xWindowsUpdateAgent

PowerShell commands for Azure DSC

> Import-AzureRmAutomationDscConfiguration
> Start-AzureRmAutomationDscCompilationJob
> Register-AzureRmAutomationDscNode
> Set-AzureRmAutomationDscNode

- Continuous control – check up to every 15 min
- Autocorrection – not only reporting
- Centralized storage and distribution of configs
- Out of the box – write once, apply to all
- Library of pre-baked configurations from MS and community

Demo: Enroll machine with DSC



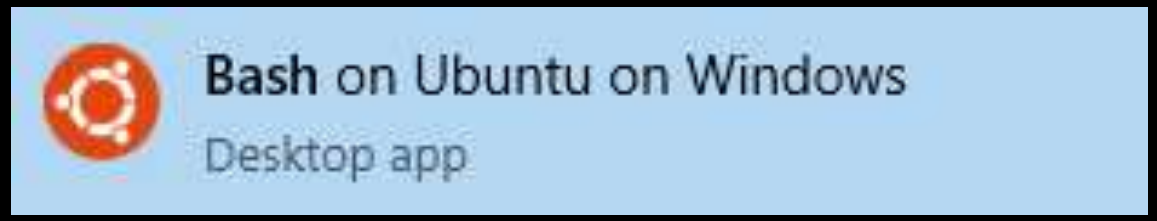
John wants to control state drift of his VM with DSC

Ansible – making changes

- Agentless and simple
- Used to be Linux-native
- Utilizes WinRM on Windows, SSH on Linux
- Requires Linux VM to control
- Has modules for EVERYTHING

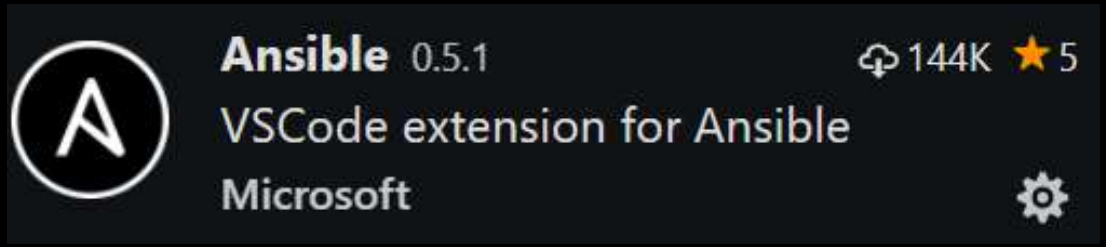
Running Ansible from “Windows”

Linux VM or Subsystem




```
igor@SE01-C6GDHC2: ~
ansible is already the newest version (2.0.0.2-2ubuntu1.1).
0 upgraded, 0 newly installed, 0 to remove and 259 not upgraded.
igor@SE01-C6GDHC2:~$ ansible
Usage: ansible <host-pattern> [options]

Options:
  -a MODULE_ARGS, --args=MODULE_ARGS
                        module arguments
  --ask-become-pass     ask for privilege escalation password
  -k, --ask-pass        ask for connection password
  --ask-su-pass         ask for su password (deprecated, use become)
  -K, --ask-sudo-pass   ask for sudo password (deprecated, use become)
  --ask-vault-pass      ask for vault password
  -B SECONDS, --background=SECONDS
                        run asynchronously, failing after X seconds
                        (default=N/A)
  -b, --become          run operations with become (nopasswd implied)
  --become-method=BECOME_METHOD
                        privilege escalation method to use (default=sudo),
                        valid choices: [ sudo | su | pbrun | pfexec | runas |
                        doas ]
  --become-user=BECOME_USER
                        run operations as this user (default=root)
  -C, --check           don't make any changes; instead, try to predict some
                        of the changes that may occur
  -c CONNECTION, --connection=CONNECTION
                        connection type to use (default=smart)
  -D, --diff            when changing (small) files and templates, show the
                        differences in those files; works great with --check
  -e EXTRA_VARS, --extra-vars=EXTRA_VARS
```

Write in VS Code




```yaml
---
- name: Install sysmon
  win_chocolatey:
    name: sysmon
    state: present
  register: sysmon_installed
  tags:
    - monitoring

- name: Create temp directory
  win_file:
    path: C:\Temp\
    state: directory
  tags:
```

Running Ansible against Windows

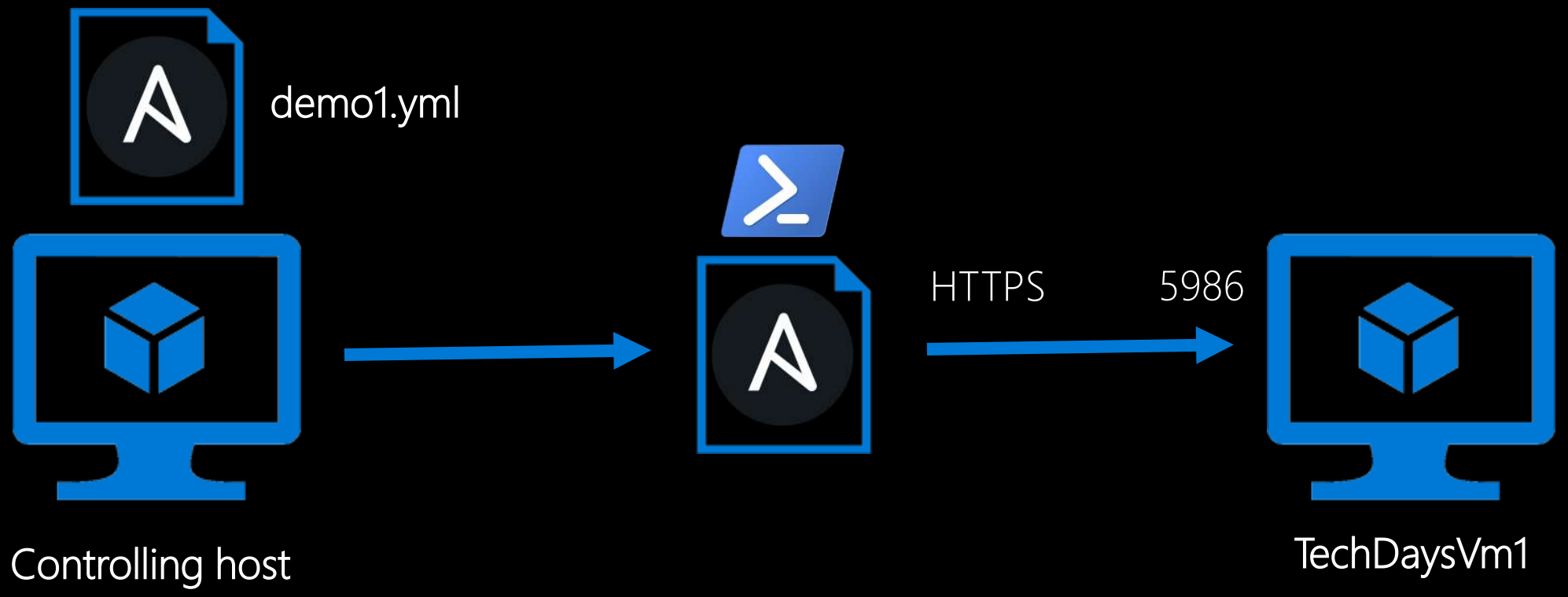
Windows 7+, Server 2008+

PS 3.0+, .NET 4.0+

Active WinRM listener
(can be configured with DSC)

How Ansible works

> ansible-playbook demo1.yml --inventory="demohosts/hosts"



YML: playbooks and tasks

```yaml
---
- hosts: TechdaysDemo
  roles:
    - Demo1
```

Target hosts from inventory

Roles to execute

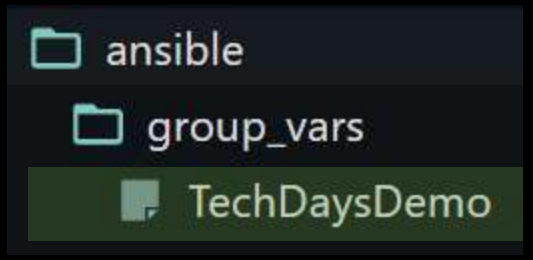
```
demohosts
  hosts
```

```
[TechdaysDemo]
techdaysvm1
techdaysvm2
techdaysvm3
135.45.61.98

[NotForDemo]
techdaysvm4
techdaysvm5
```

```
roles
  Demo1
    tasks
      main.yml
```

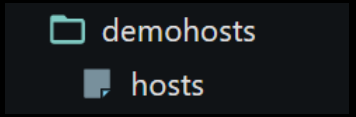
```yaml
---
- name: Install sysmon
  win_chocolatey:
    name: sysmon
    state: present
  register: sysmon_installed
  tags:
    - monitoring

- name: Create temp directory
  win_file:
    path: C:\Temp\
    state: directory
  tags:
```

Connection settings



Variables for host group

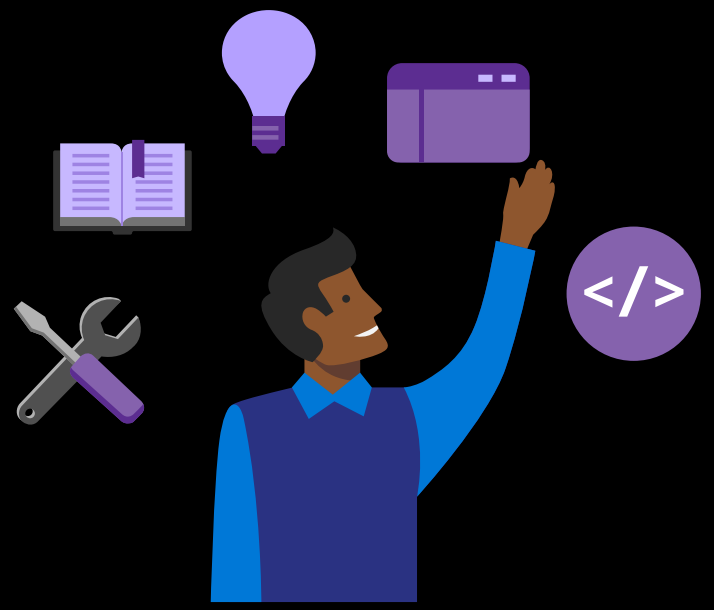
```
ansible_user: techdaysuser
ansible_password: ClewUB9wFuznae92eqyUVIwevP2Znz
ansible_connection: winrm
ansible_port: 5986
ansible_winrm_server_cert_validation: ignore
```




```
[TechdaysDemo]
techdaysvm1
techdaysvm2
techdaysvm3
135.45.61.98

[NotForDemo]
techdaysvm4
techdaysvm5
```

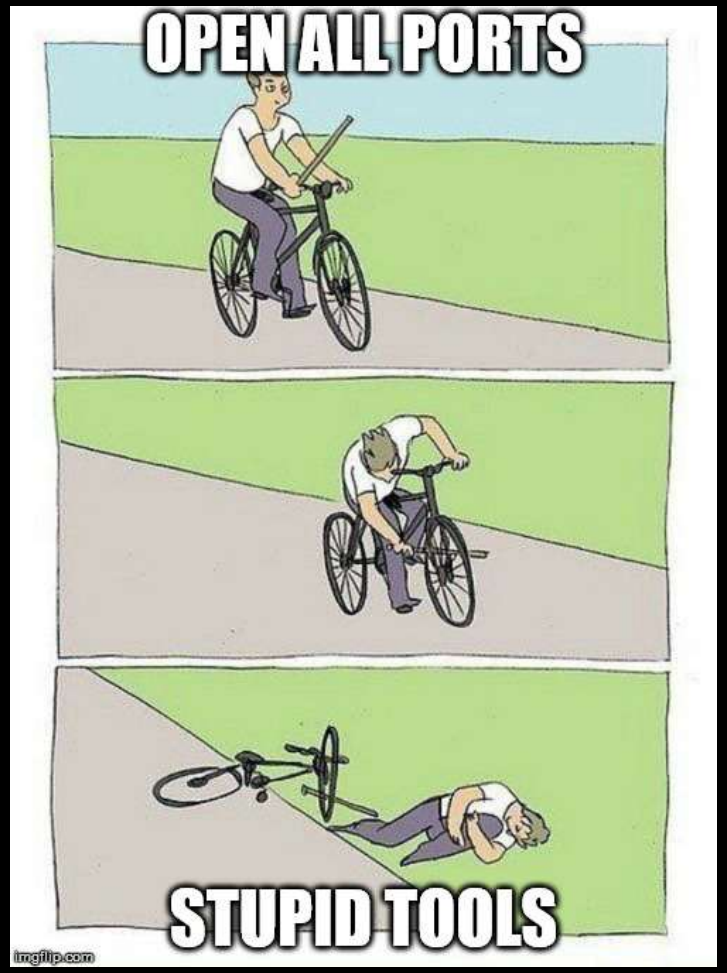
```
PLAY [TechdaysDemo] ***********************************************************

TASK [Gathering Facts] ********************************************************
ok: [40.127.165.40]

TASK [Demo1 : Install sysmon] *************************************************
changed: [40.127.165.40]

TASK [Demo1 : Create temp directory] ******************************************
changed: [40.127.165.40]

TASK [Demo1 : Download SwiftOnSecurity sysmon conf] ***************************
changed: [40.127.165.40]

TASK [Demo1 : Install new sysmon configuration] *******************************
changed: [40.127.165.40]

TASK [Demo1 : Activate new sysmon configuration] ******************************
changed: [40.127.165.40]

PLAY RECAP ********************************************************************
40.127.165.40              : ok=6    changed=5    unreachable=0    failed=0
```

Demo: Making changes with Ansible



John wants to install Sysmon with hardened config to get better security logs

- Don’t weaken perimeter security by opening ALL ports for tools
- Limit connection IP ranges
- Use WinRM over HTTPs
- Encrypt passwords with certificates (DSC) / vault (Ansible)
- Don’t store plaintext secrets in code



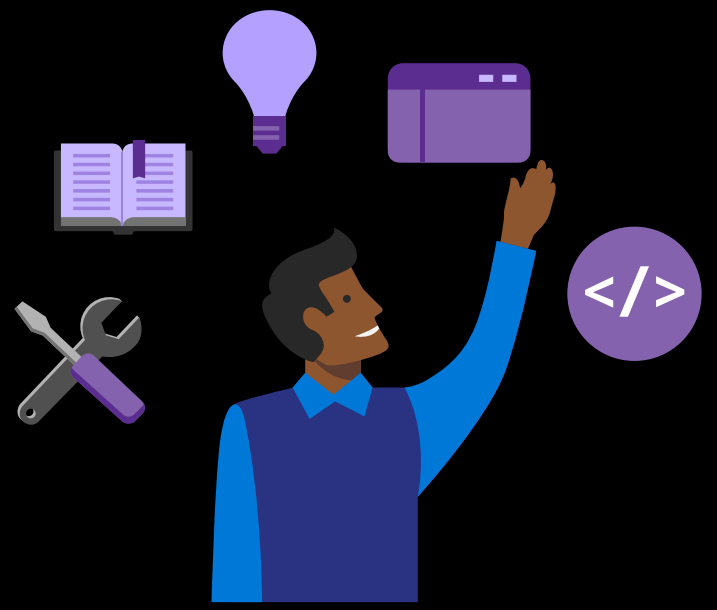
Adding more security as 1-2-3

- Move secrets to Vault; WinRM over HTTPS
- Assign Azure policy for security
- Apply Windows baseline security configuration
- Send message to all active users

OS baseline hardening

Based on GPO recommendations of DoD DISA STIG from 22/06/18

Demo: Securing resources



The company is under attack of hackers

(adding 80% of value with 20% of effort)

Alternatives

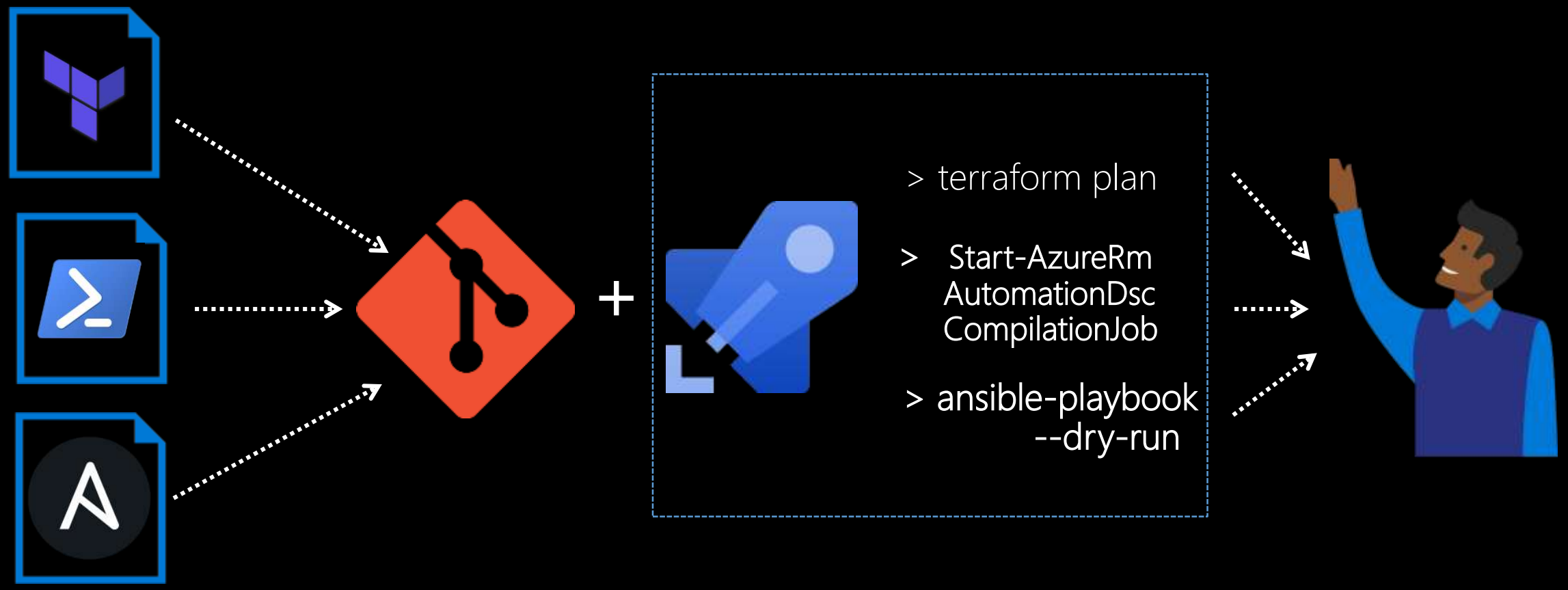
Terraform – ARM - Ansible

ARM + Terraform provider (preview)

Azure DSC - Ansible

Ansible - PowerShell

CI/CD for infrastructure code



CI/CD for infrastructure code

Git everything

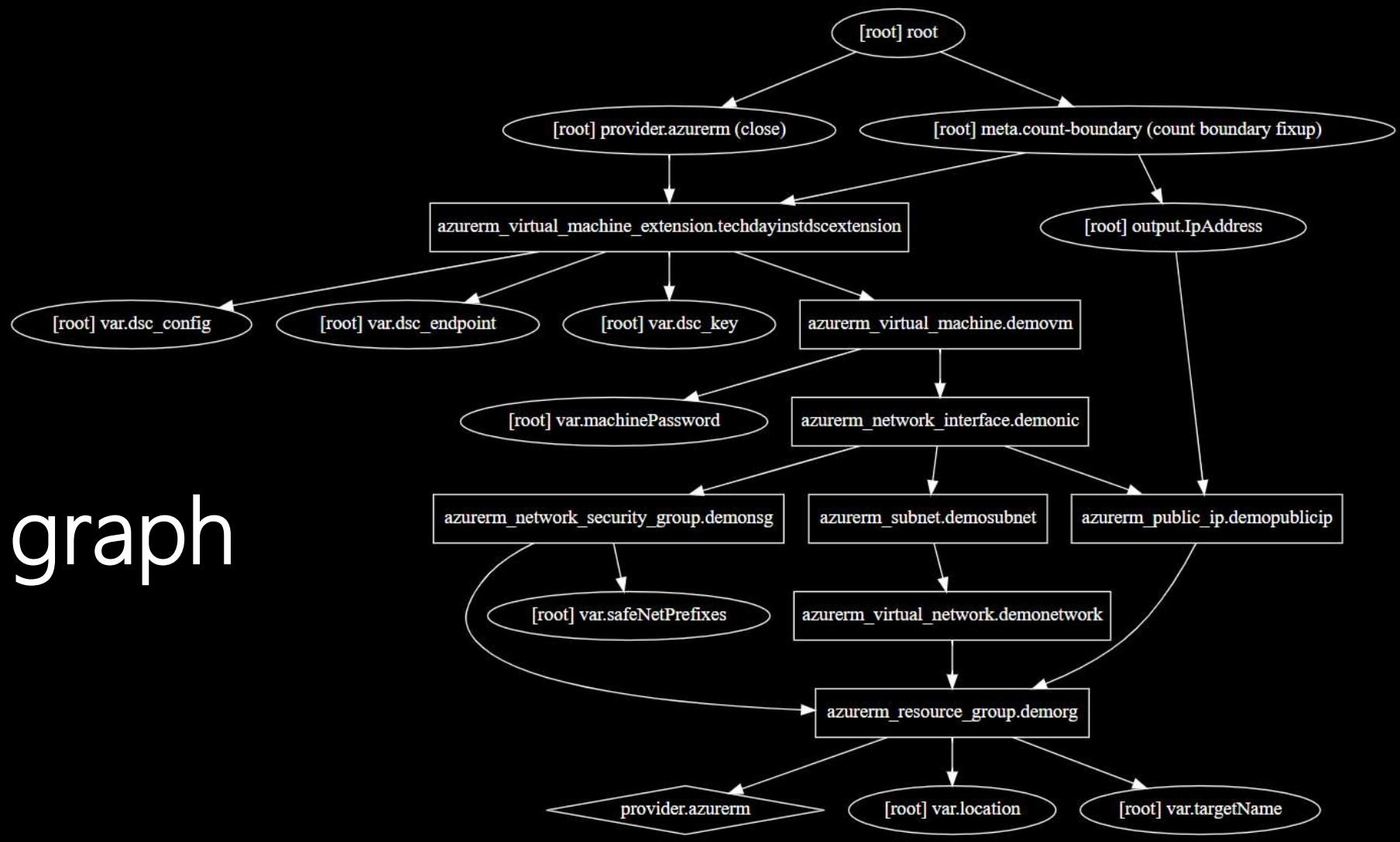
Automatic *terraform plan* for templates

Verify DSC configuration by applying it

Ansible dry-run

Practice recovery drills

Bonus: Infrastructure graph




```
> terraform graph | dot.exe -Tsvg > graph.svg
```

Control
Visibility
Automation

20% time / 80% value

80% time / 20% value

Thank you!

Please evaluate my session in the TechDays app!