vmworld

POSSIBLE BEGINS WITH YOU

MGT3711BUS

A Tale of CI/CD, Infrastructure as Code, and Data Protection

Rebecca Fitzhugh

Jaap Brasser

#VMworld #MGT3711BUS

vmware

Rebecca Fitzhugh

- @RebeccaFitzhugh
- technicloud.com
- https://github.com/rfitzhugh
- vBrownBag
- vSphere Virtual Machine Management
  Learning VMware vSphere
- VCDX #243

Jaap Brasser

- @jaap_brasser
- jaapbrasser.com
- https://github.com/jaapbrasser
- Cloud and Datacenter Management

What is Infrastructure as Code?

A **practice** in which infrastructure automation is based on practices from software development.

Emphasizes **consistent**, **repeatable** routines for **provisioning** and **modifying** systems and their configuration.

Changes are made to **definitions** and then rolled out to systems through **unattended processes** that include thorough **validation**.

Why Infrastructure as Code?

Speed
Infrastructure as code systematizes the process shaving off time, an important factor in today's DevOps environments.

Efficiency
All the instructions needed, including the relationships between computers and networks, are entered already, making the process much more seamless and efficient.

Lowering Risk
Significantly reduces risk of human error. Avoid typing instructions over and over again, or having developers forget to include certain machines in the loop.

What is CI/CD?

CI/CD refers to the combined practices of **continuous integration** and **continuous delivery**.

Emphasizes **test automation** to maintain **consistency** as frequent changes are merged.

After changes are made, release processes are **automated** to quickly **deploy** new updates and versions.

Why CI/CD?

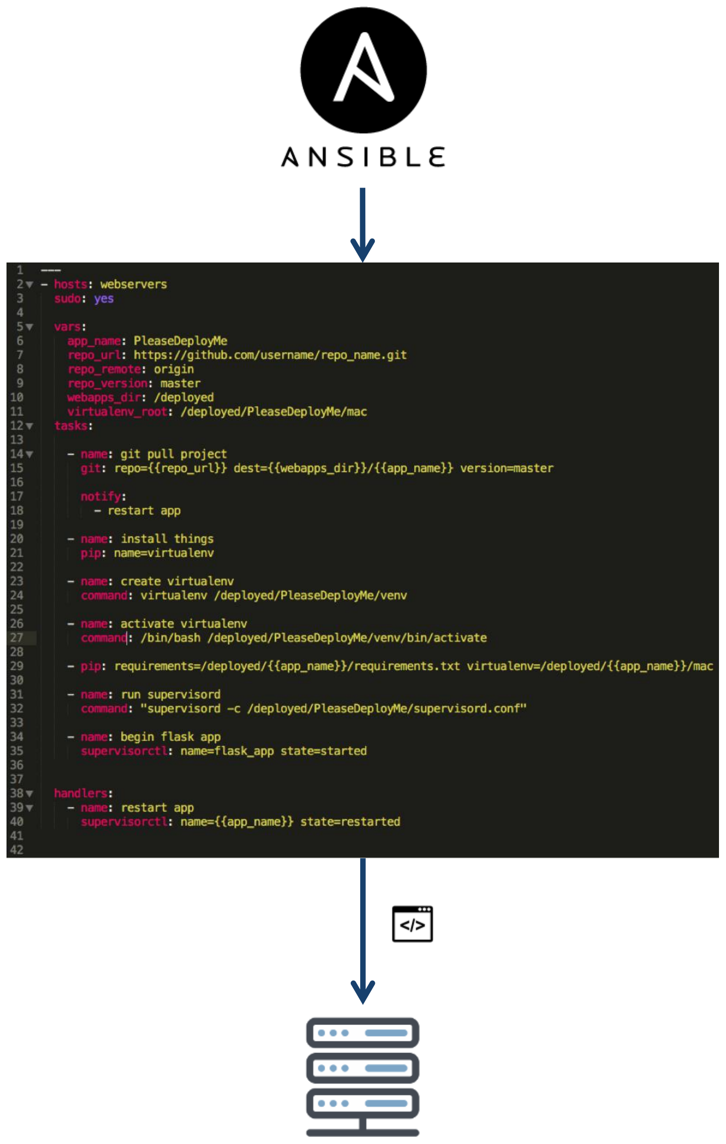
Quick Detection
Problems are found more quickly, and therefore can be worked at while small before problems are compounded. Fix issues while fresh.

Improved Integration
Automating testing increases reliability as there is a consistent test environment. Quicker detection allows more time for integration testing.

Lowering Risk
Releasing more frequently reduces the surface area of each release thus lowering risk. The goal is to make releases “boring”. Get fast feedback from users.

Configuration Management Tools

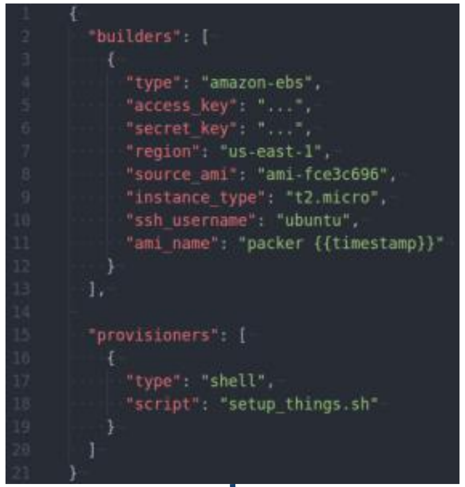
- Chef, Puppet, Ansible, SaltStack are all configuration management tools; designed to install and manage software on existing servers.
- **Coding conventions** – consistent and predictable structure, file layout, clearly named parameters, secrets management, etc.
- **Idempotent code** – continuously executing the same code repeatedly while providing the same result.



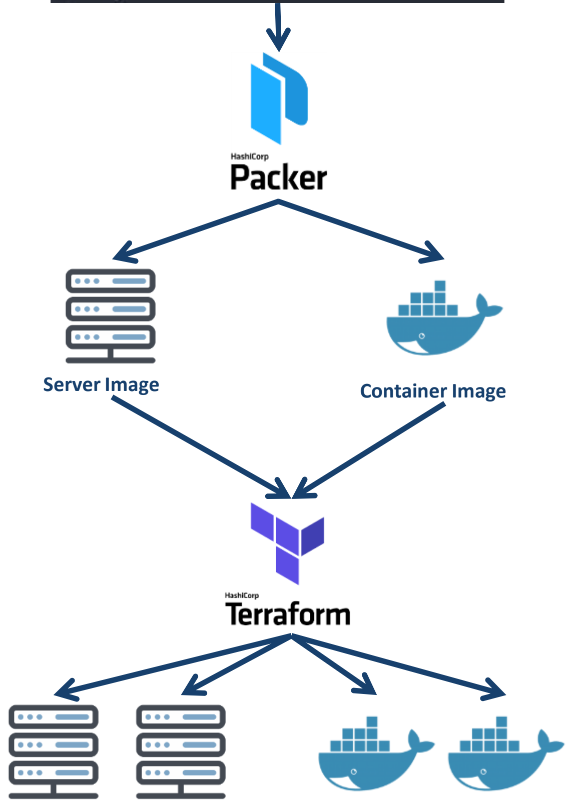
```yaml
---
- hosts: webservers
  sudo: yes

  vars:
    app_name: PleaseDeployMe
    repo_url: https://github.com/username/repo_name.git
    repo_remote: origin
    repo_version: master
    webapps_dir: /deployed
    virtualenv_root: /deployed/PleaseDeployMe/mac
  tasks:

    - name: git pull project
      git: repo={{repo_url}} dest={{webapps_dir}}/{{app_name}} version=master

      notify:
        - restart app

    - name: install things
      pip: name=virtualenv

    - name: create virtualenv
      command: virtualenv /deployed/PleaseDeployMe/venv

    - name: activate virtualenv
      command: /bin/bash /deployed/PleaseDeployMe/venv/bin/activate

    - pip: requirements=/deployed/{{app_name}}/requirements.txt virtualenv=/deployed/{{app_name}}/mac

    - name: run supervisord
      command: "supervisord -c /deployed/PleaseDeployMe/supervisord.conf"

    - name: begin flask app
      supervisorctl: name=flask_app state=started


  handlers:
    - name: restart app
      supervisorctl: name={{app_name}} state=restarted

```

Server Templating Tools

- Docker, Packer, and Vagrant
- Create an image of a server that captures a fully self-contained “snapshot” of the OS, software, files, and other details.
- And move on to the deployment step in the pipeline…

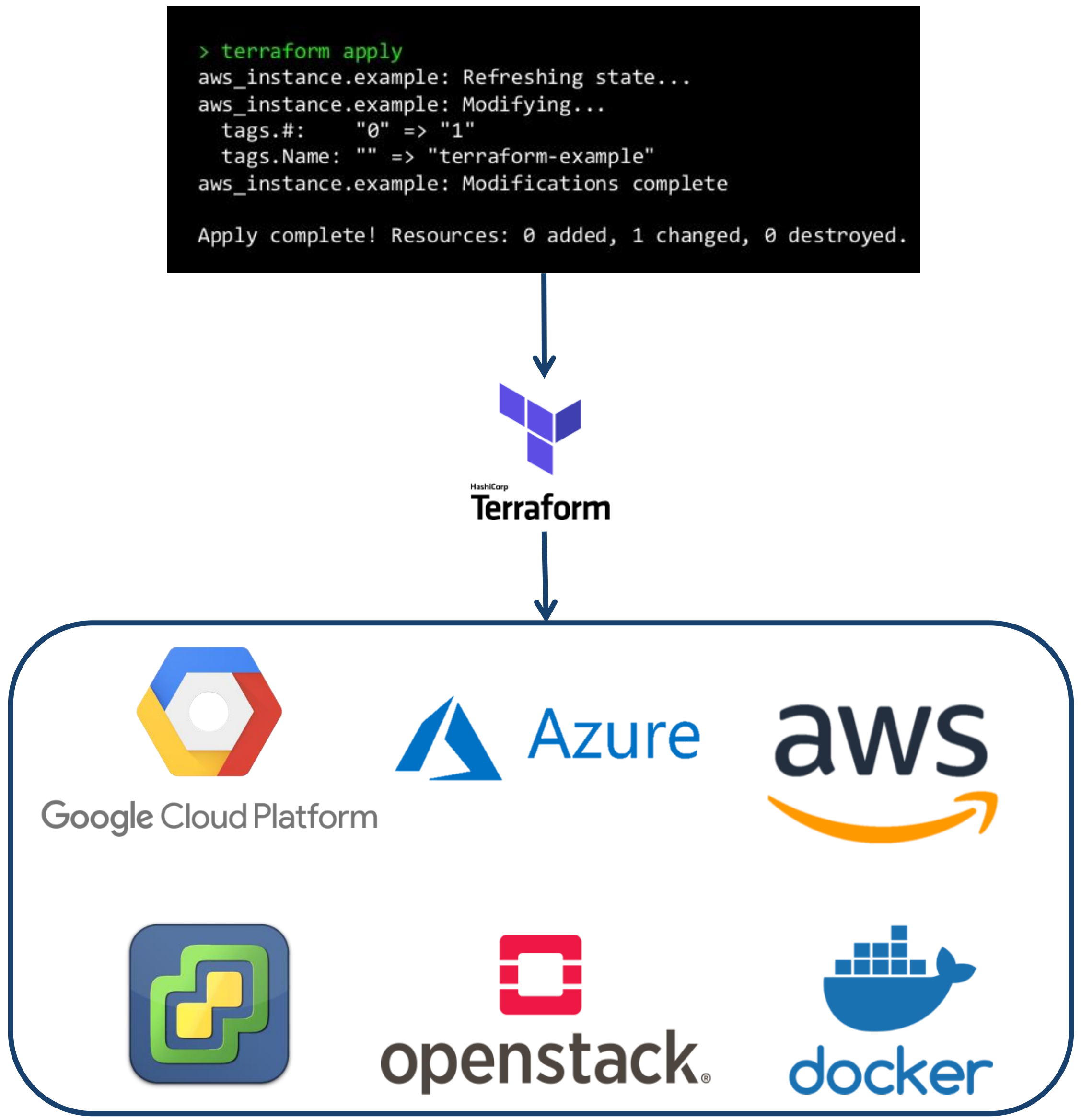


```
{
  "builders": [
    {
      "type": "amazon-ebs",
      "access_key": "...",
      "secret_key": "...",
      "region": "us-east-1",
      "source_ami": "ami-fce3c696",
      "instance_type": "t2.micro",
      "ssh_username": "ubuntu",
      "ami_name": "packer {{timestamp}}"
    }
  ],

  "provisioners": [
    {
      "type": "shell",
      "script": "setup_things.sh"
    }
  ]
}
```



Provisioning Tools

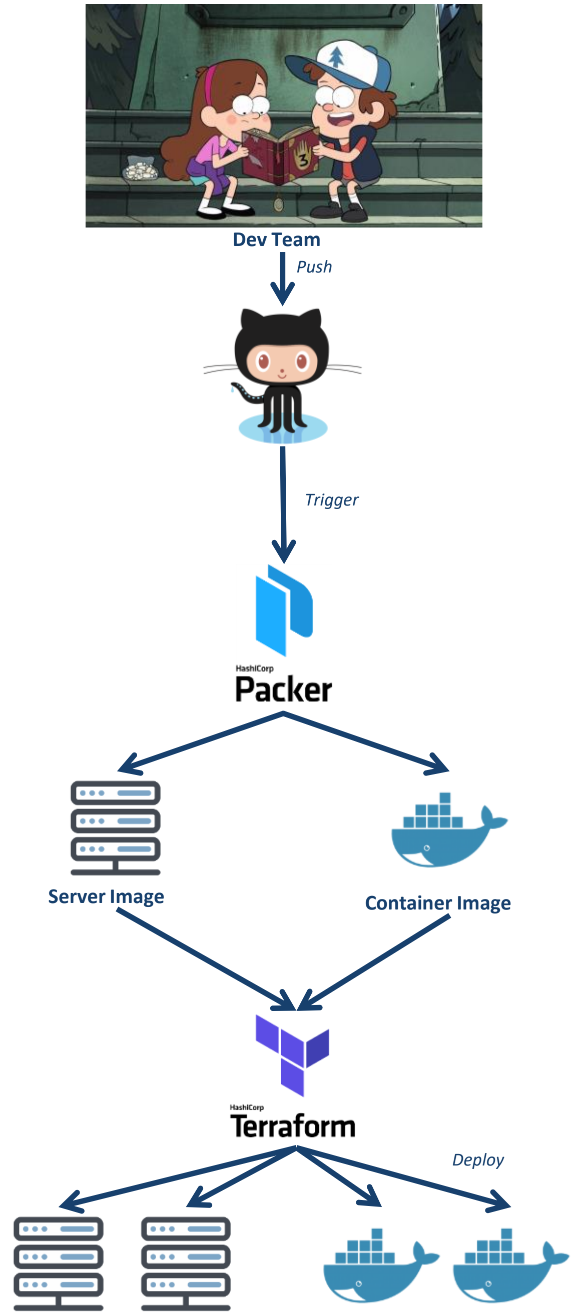
- Terraform, Azure Resource Manager, AWS CloudFormation, and OpenStack Heat create servers.
- Use this tool to not only create servers, but also other resources such as databases, load balancers, firewall settings, storage, etc.
- Multi-Platform



```
> terraform apply
aws_instance.example: Refreshing state...
aws_instance.example: Modifying...
  tags.#:     "0" => "1"
  tags.Name: "" => "terraform-example"
aws_instance.example: Modifications complete

Apply complete! Resources: 0 added, 1 changed, 0 destroyed.
```

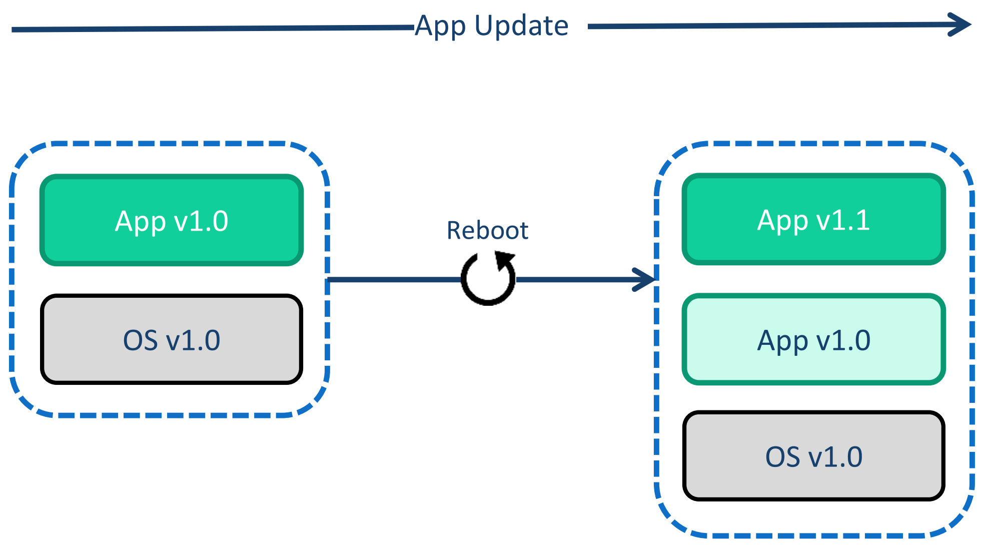
Version (Source) Control

- A system that records changes to a file or set of files over time you that specific **versions** can be recalled later.
- Complete visibility and auditing of **all** changes.



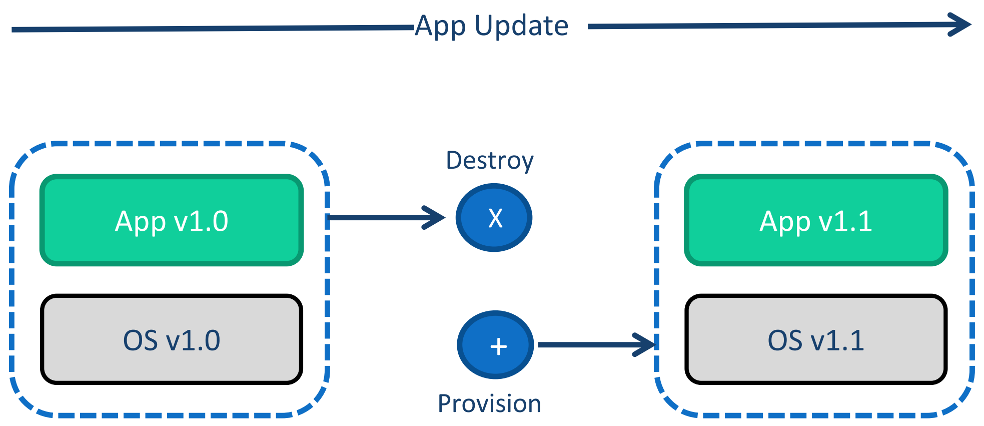
Mutable Infrastructure

- Infrastructure is continually updated, patched, and tuned to meet ongoing needs.
- CM tools, such as Chef or Puppet, typically default to mutable infrastructure paradigm.
- Over time, as more patches are applied, each server builds up a unique history of changes.
- As a result, each server is slightly different than the others, leading to configuration drift.



Immutable Infrastructure

- All changes result in deployment of new app version.
- Reduces likelihood of configuration drift.
- Automated testing is more effective; an immutable image that passes all tests is likely to behave the same in production.
- Blue / green



Imperative Programming

- Procedural
- Defines specific commands that need to be executed in the appropriate order to end with the desired conclusion.
- AKA “the directions”

5

Shack is #2 on the right side.

1

Leave the house

2

Get in the car

3

Drive straight on Falls Road for 5 miles.

4

Turn left on Gravity Ave and drive for 2 blocks.

Declarative Programming

- Functional
- Defines the desired state and the system executes what is needed to achieve the desired state.
- AKA “the destination”

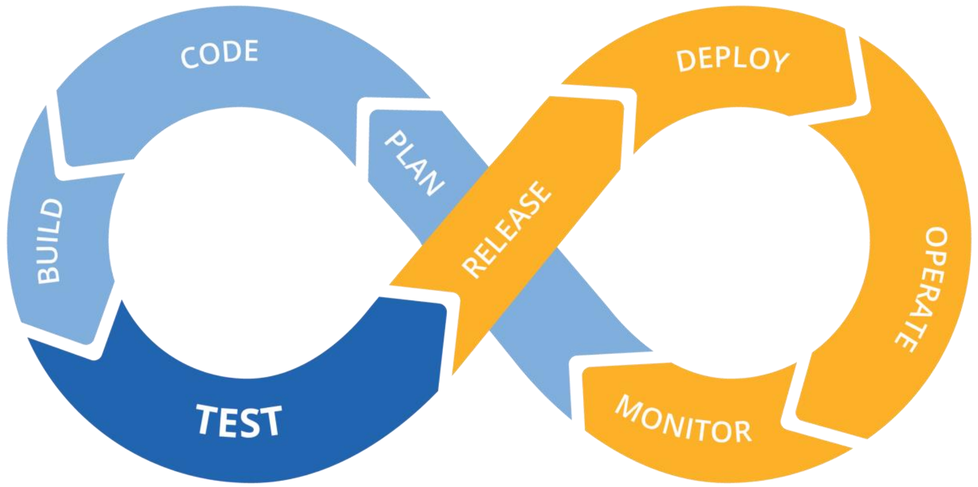
The address is:

2 Gravity Ave

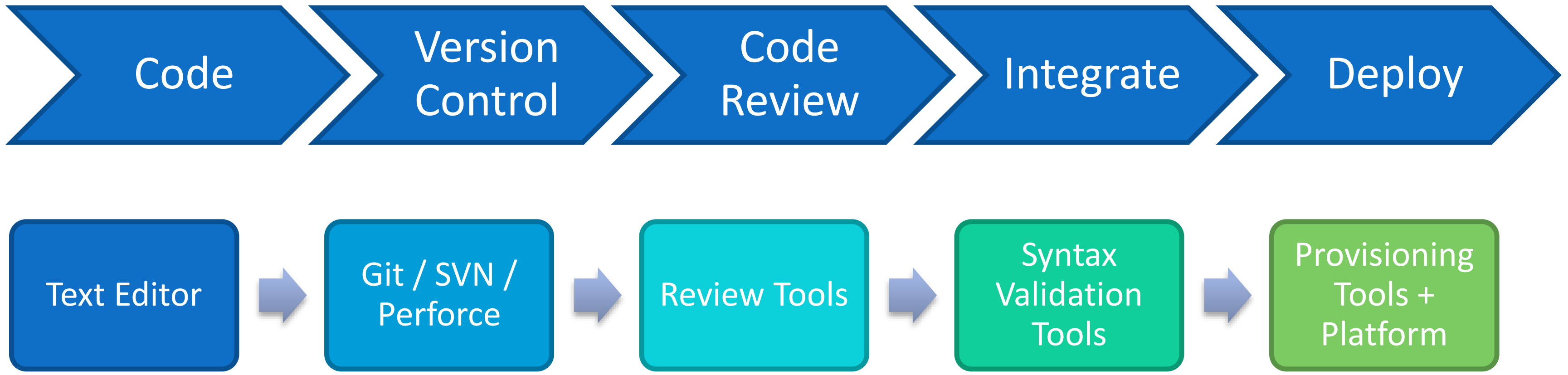
Gravity Falls, MI 29014

Principles

- All your stuff can be **rebuilt**
- All your things are **disposable**
- Everything is **consistent**
- There is no **end state**

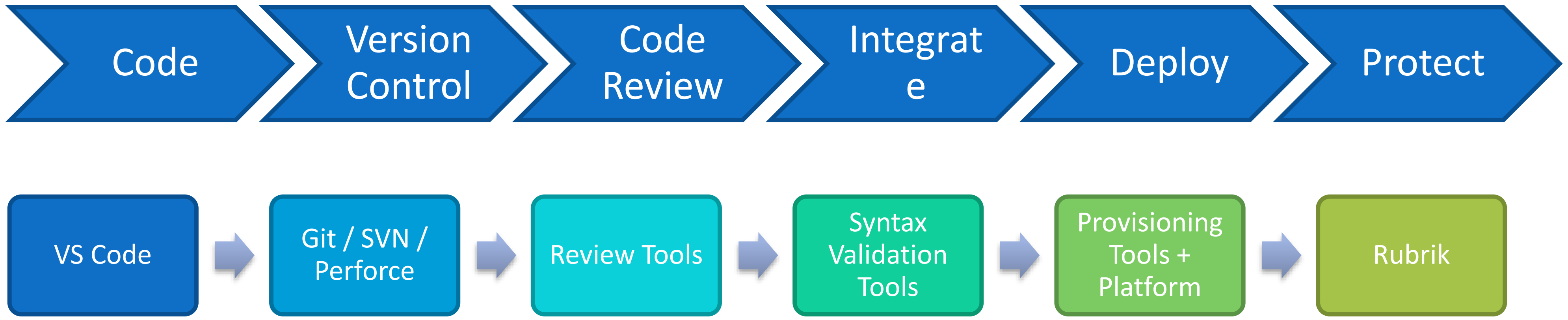


Basic Workflow



It's all software.

Basic Workflow + Day 2



It’s all software.

Critical Design Goals

Simplicity

Reducing risk and crafting a simplified architecture is the hallmark of excellence.

Scalability

Meeting design requirements with an understanding that change is constant and the future demands a dynamic architecture.

Automation

Consistent, reliable, and repeatable processes make automation for self-healing and autonomous systems a reality.

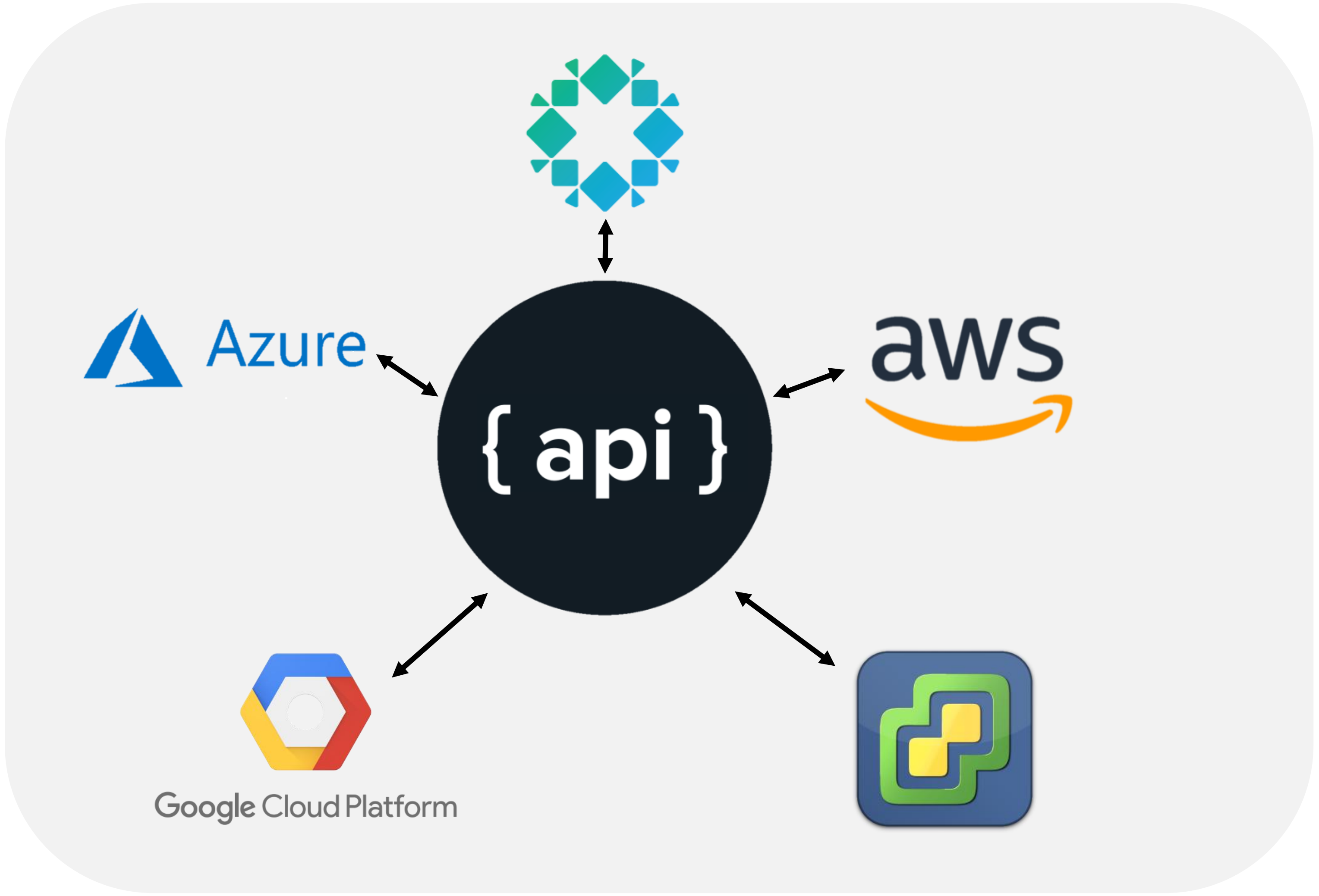
Demo!

Automated VM Deployment

Day 2 and beyond!

Modern Data Protection Use Cases

Creating a Software-Defined Architecture



Common Customer Use Cases

1 **Cloud Management Portal Integration**

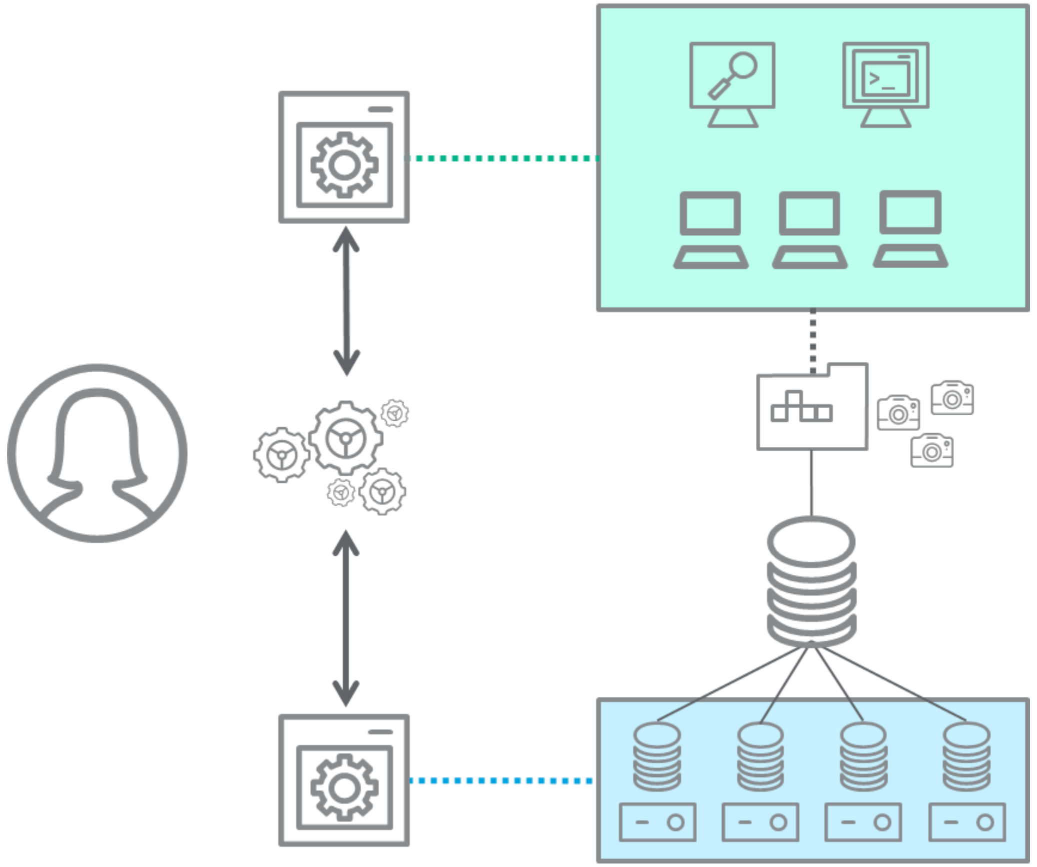
Leverage dashboards, workflows, and service catalogues from Service Now, VMware vRealize Suite, and others.

2 **Configuration Management**

Use your existing investment in desired state tools to protect and restore data at a global scale across the Rubrik fabric.

3 **Application and Server Validation**

Create live copies in near-zero timeframes to test restoration of data, applications, and entire servers against an orchestration engine.



Demo!

Backup Validation

Backup Validation

How do I know I can *actually* restore from my backups?

Workflow Architecture

STEP ONE
STAGE

Bring up all sessions and Live Mounts required to begin the battery of tests to be performed

STEP TWO
TEST

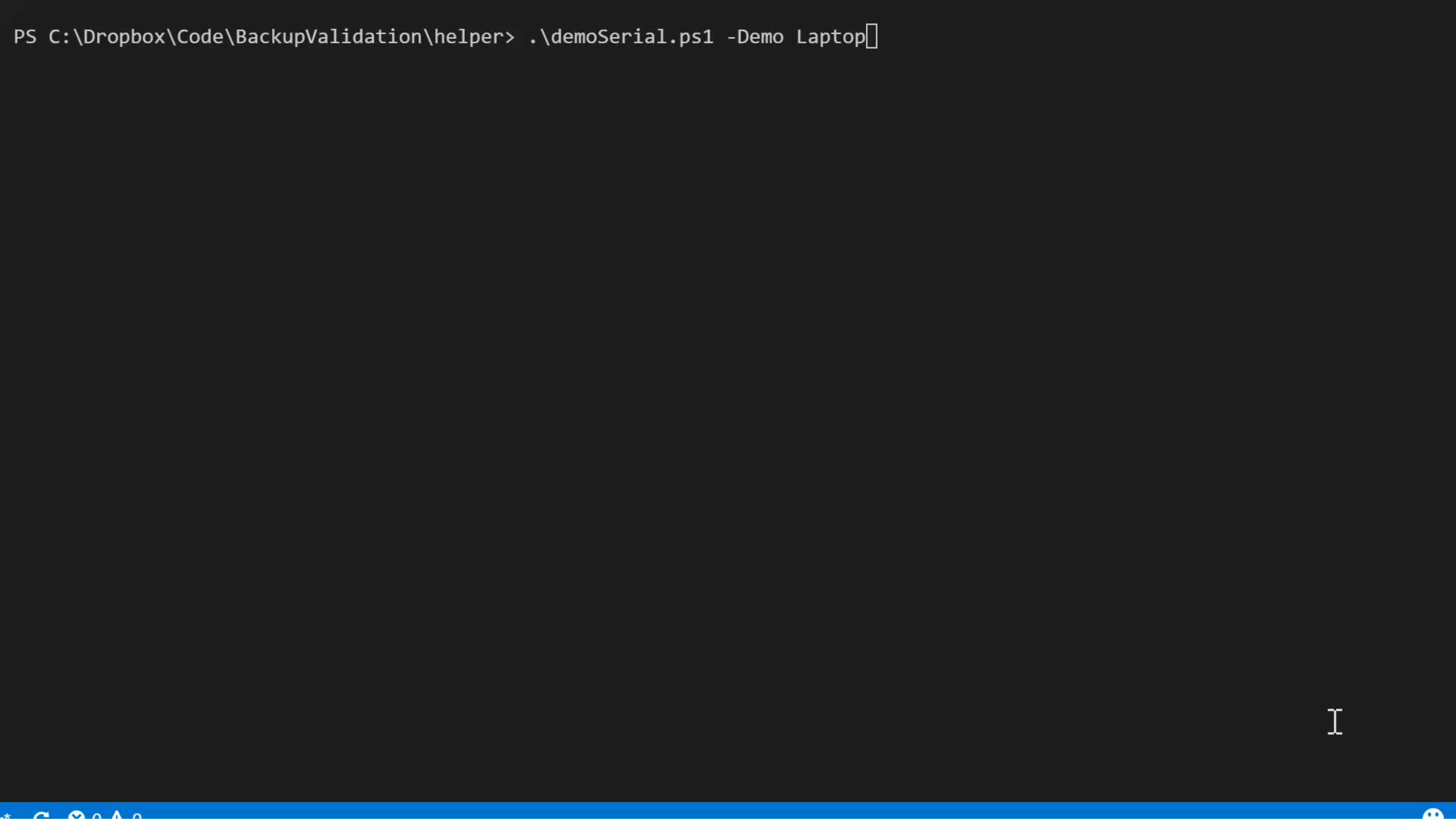
Validate all of the network, service, application, and data requirements are in play

STEP THREE
CLEANUP

Remove all artifacts from the environment, often replaying the steps in reverse

STEP FOUR
REPORT

Store data on the workflow and, if required, remediate issues that were found



```
PS C:\Dropbox\Code\BackupValidation\helper> .\demoSerial.ps1 -Demo Laptop
```

Workflow Architecture – Today's Example

STEP ONE
STAGE

- ✓ Connect to vCenter
- ✓ Connect to Rubrik
- ✓ Request a Live Mount
- ✓ Network Setup

STEP TWO
TEST

- ✓ Network Ping
- ✓ Service Validation
- ✓ Application Query

STEP THREE
CLEANUP

- ✓ Delete Live Mount
- ✓ Release IP Address

STEP FOUR
REPORT

- ✓ Send Test Report
- ✓ Email on Drift

Staging Tasks

STEP ONE

STAGE

- ✓ Connect to vCenter
- ✓ Connect to Rubrik
- ✓ Request a Live Mount
- ✓ Network Setup

Establish Sessions with Required Endpoints

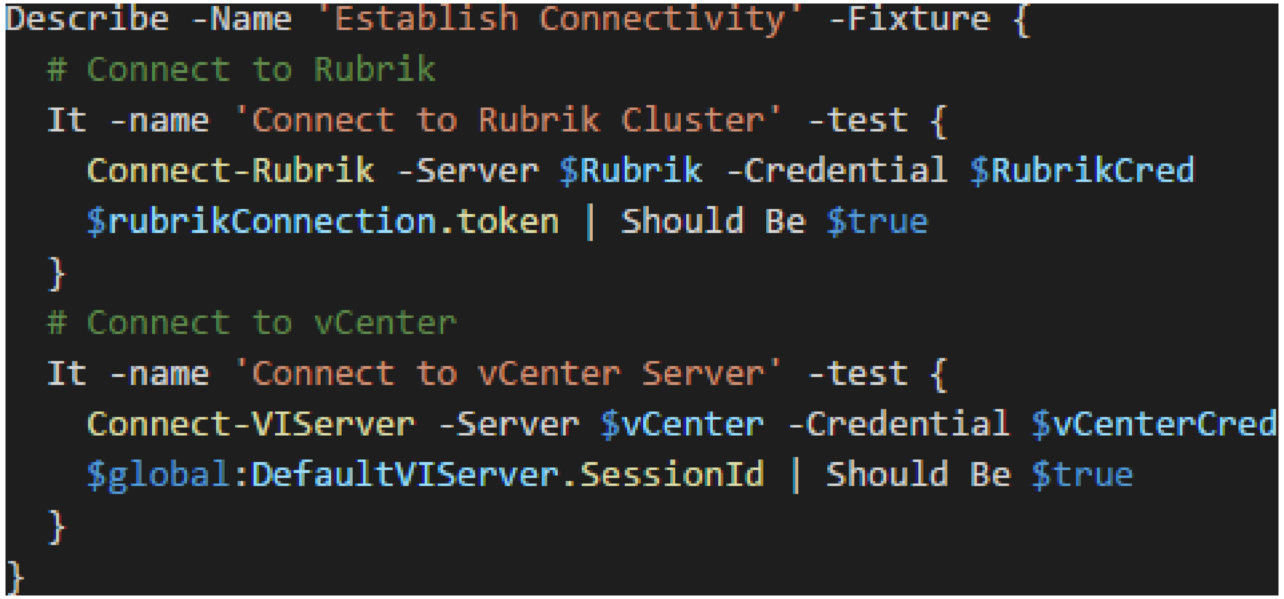
Sessions are authenticated relationships with management endpoints needed to perform work. In this case, Rubrik CDM and VMware vSphere are being asked to execute tasks on behalf of the workflow.

Build a Live Mount

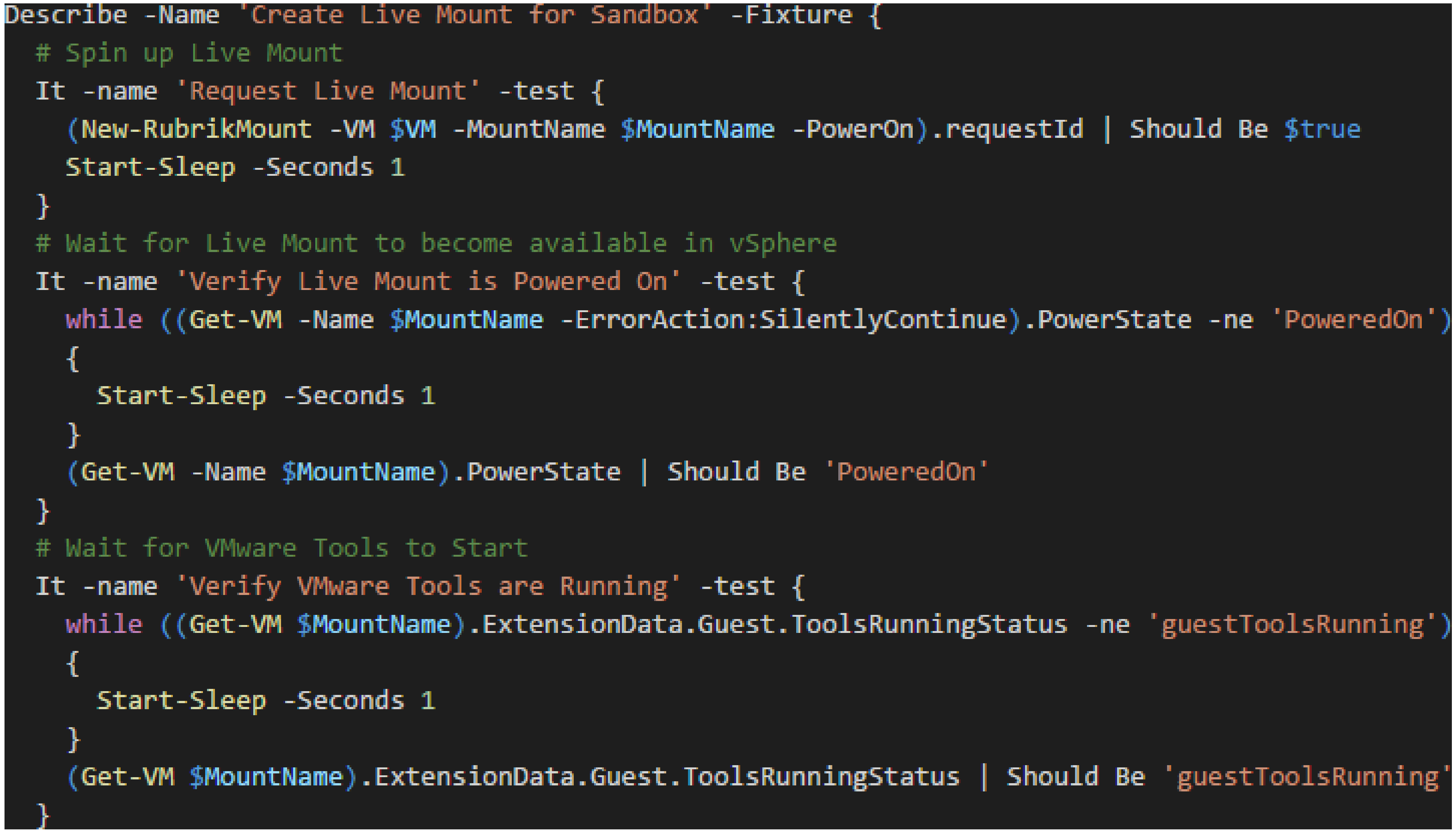
Rubrik’s Live Mount capability instantiates an entire virtual machine from any backup, which we call a snapshot, in a matter of seconds. This has zero impact on the production workload or backup data.

Network Configuration and IP Address Settings

An isolated “black hole” network will be used to allow a complex set of workloads to be run and talk to one another. This is commonly a VLAN behind a proxy, NAT device, or flat network with a dual-homed test server.



```
Describe -Name 'Establish Connectivity' -Fixture {
  # Connect to Rubrik
  It -name 'Connect to Rubrik Cluster' -test {
    Connect-Rubrik -Server $Rubrik -Credential $RubrikCred
    $rubrikConnection.token | Should Be $true
  }
  # Connect to vCenter
  It -name 'Connect to vCenter Server' -test {
    Connect-VIServer -Server $vCenter -Credential $vCenterCred
    $global:DefaultVIServer.SessionId | Should Be $true
  }
}
```



```
Describe -Name 'Create Live Mount for Sandbox' -Fixture {
  # Spin up Live Mount
  It -name 'Request Live Mount' -test {
    (New-RubrikMount -VM $VM -MountName $MountName -PowerOn).requestId | Should Be $true
    Start-Sleep -Seconds 1
  }
  # Wait for Live Mount to become available in vSphere
  It -name 'Verify Live Mount is Powered On' -test {
    while ((Get-VM -Name $MountName -ErrorAction:SilentlyContinue).PowerState -ne 'PoweredOn')
    {
      Start-Sleep -Seconds 1
    }
    (Get-VM -Name $MountName).PowerState | Should Be 'PoweredOn'
  }
  # Wait for VMware Tools to Start
  It -name 'Verify VMware Tools are Running' -test {
    while ((Get-VM $MountName).ExtensionData.Guest.ToolsRunningStatus -ne 'guestToolsRunning')
    {
      Start-Sleep -Seconds 1
    }
    (Get-VM $MountName).ExtensionData.Guest.ToolsRunningStatus | Should Be 'guestToolsRunning'
  }
```

Testing Tasks

STEP TWO

TEST

- ✓ Network Ping
- ✓ Service Validation
- ✓ Application Query

Network Ping

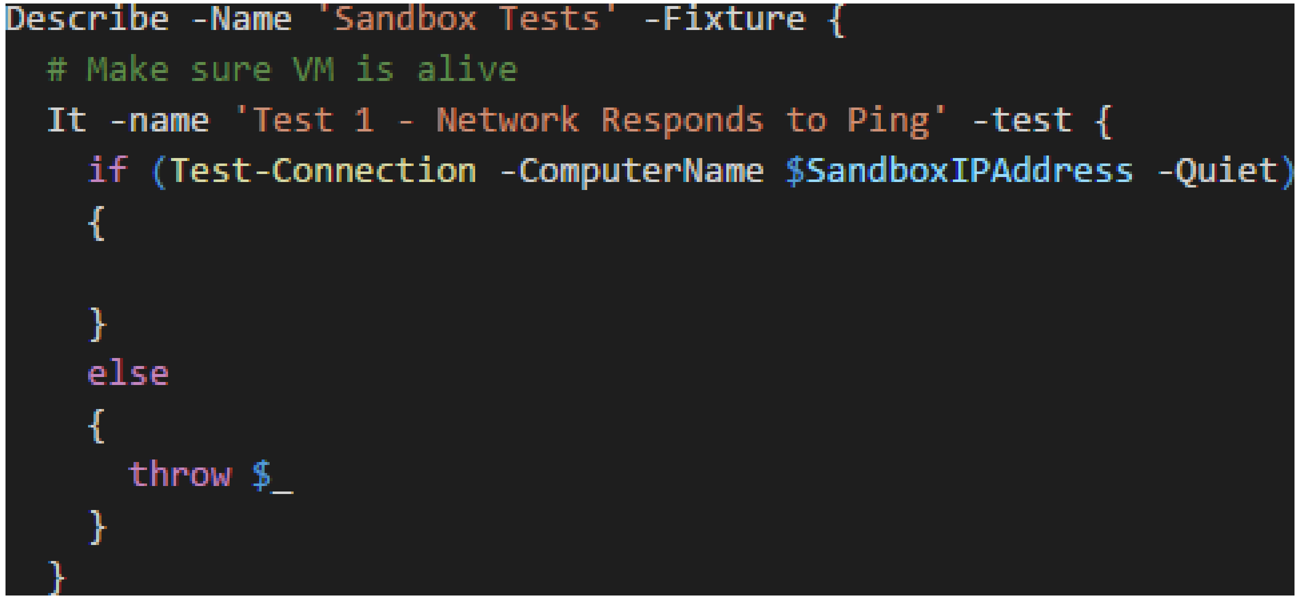
Before using advanced testing, it's wise to make sure that the server is reachable with a simple ping test. This assumes that **ICMP Echo** is allowed on the test network.

Service Validation

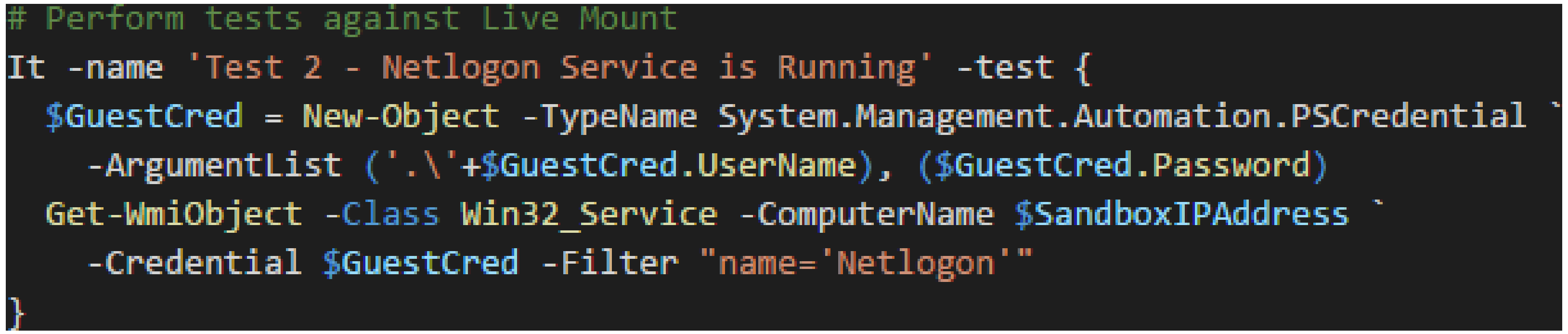
For many use cases, simply making sure that the proper services or daemons are running and reachable is enough to provide evidence against partition corruption, filesystem issues, dependency failures, and other single-stack problems.

Application Query

The holy grail of testing is to validate that the application stack is alive. This could be a single workload or combination of several workloads. By querying the application for data, full end-to-end testing has been performed.



```
Describe -Name 'Sandbox Tests' -Fixture {
  # Make sure VM is alive
  It -name 'Test 1 - Network Responds to Ping' -test {
    if (Test-Connection -ComputerName $SandboxIPAddress -Quiet)
    {

    }
    else
    {
      throw $_
    }
  }
```



```
# Perform tests against Live Mount
It -name 'Test 2 - Netlogon Service is Running' -test {
  $GuestCred = New-Object -TypeName System.Management.Automation.PSCredential `
    -ArgumentList ('.\'+$GuestCred.UserName), ($GuestCred.Password)
  Get-WmiObject -Class Win32_Service -ComputerName $SandboxIPAddress `
    -Credential $GuestCred -Filter "name='Netlogon'"
}
```

Cleanup Tasks

STEP THREE

CLEANUP

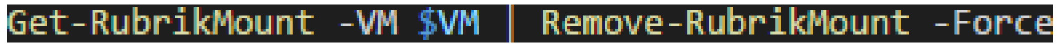
- ✓ Delete Live Mount
- ✓ Release IP Address

Live Mount Removal

A request to Rubrik CDM to remove the Live Mount will result in the virtual machine being powered off, removed from inventory, and all changes being discarded from the cluster.

Release IP Address

The test IP address is placed back into the pool of available test addresses for any other workload to consume.



```
Get-RubrikMount -VM $VM | Remove-RubrikMount -Force
```

Reporting Tasks

STEP FOUR
REPORT

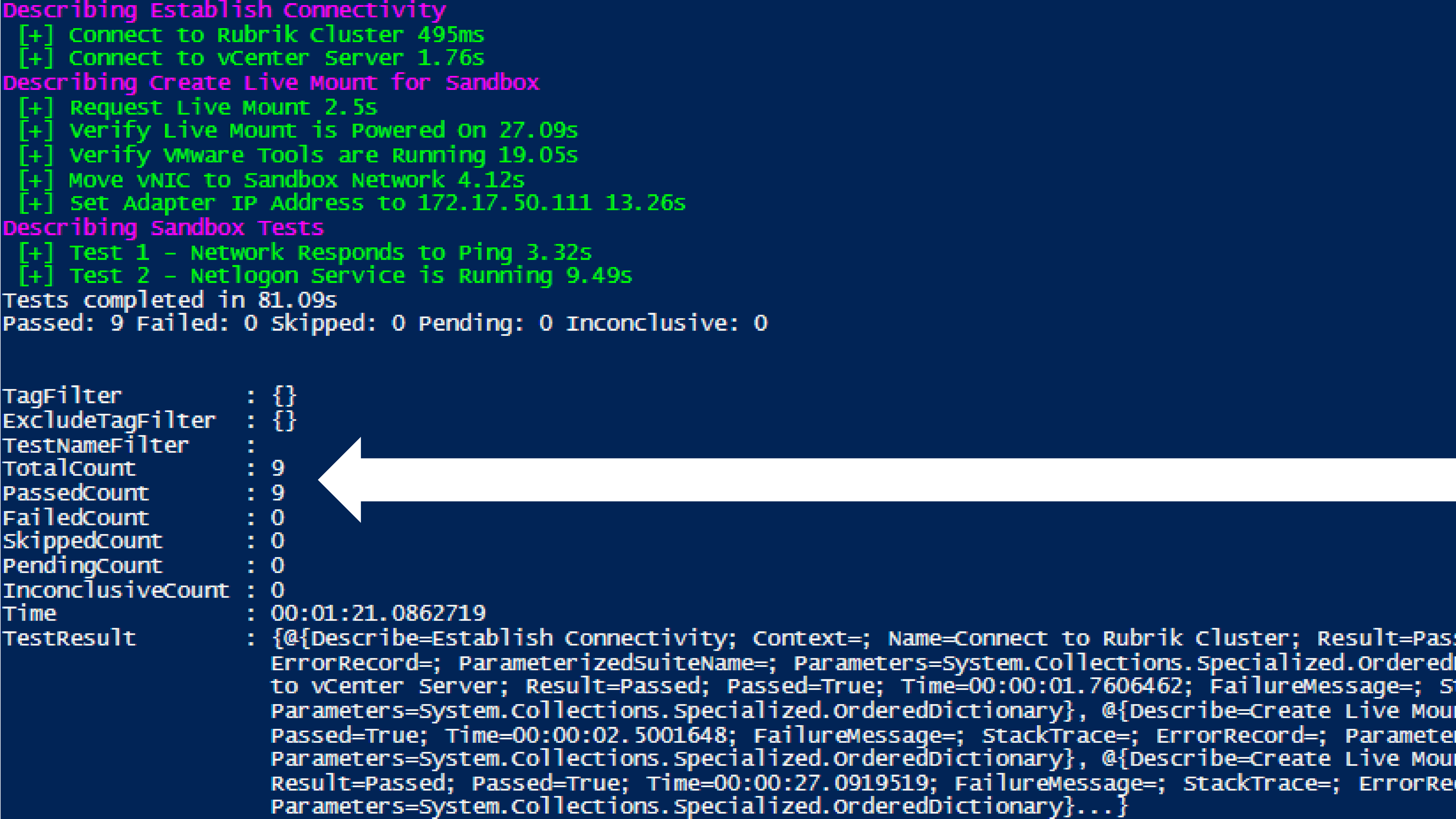
- ✓ Send Test Report
- ✓ Email on Drift

Save Reporting Data

It's important to report when a test fails due to configuration drift. If this is a new system, it may be a good idea to send an email, post a Slack message, or create a ticket. Having a human operator involved to enhance the workflow builds a body of knowledge.

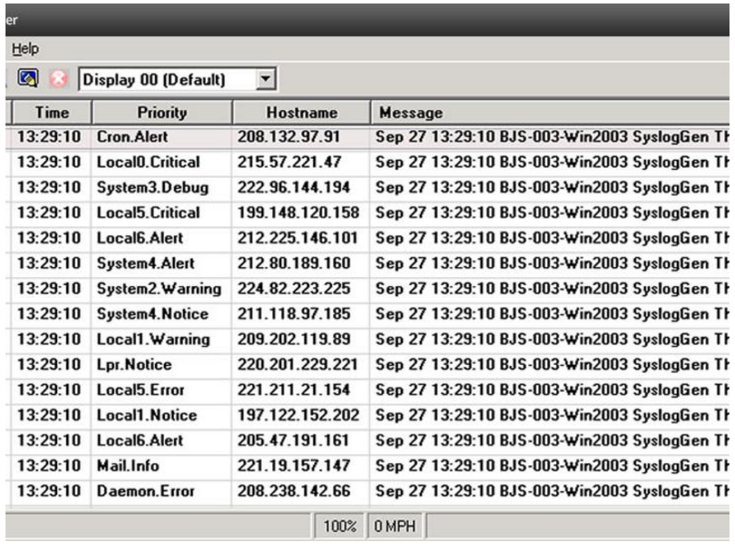
Reporting Drift

Each task – from staging, testing, and final cleanup – should be reported to a system that is collecting metrics and telemetry. This aids in remediation of workflow issues, failed tests, and helps in building new tests.



```
Describing Establish Connectivity
 [+] Connect to Rubrik Cluster 495ms
 [+] Connect to vCenter Server 1.76s
Describing Create Live Mount for Sandbox
 [+] Request Live Mount 2.5s
 [+] Verify Live Mount is Powered On 27.09s
 [+] Verify VMware Tools are Running 19.05s
 [+] Move vNIC to Sandbox Network 4.12s
 [+] Set Adapter IP Address to 172.17.50.111 13.26s
Describing Sandbox Tests
 [+] Test 1 - Network Responds to Ping 3.32s
 [+] Test 2 - Netlogon Service is Running 9.49s
Tests completed in 81.09s
Passed: 9 Failed: 0 Skipped: 0 Pending: 0 Inconclusive: 0


TagFilter         : {}
ExcludeTagFilter  : {}
TestNameFilter    :
TotalCount        : 9
PassedCount       : 9
FailedCount       : 0
SkippedCount      : 0
PendingCount      : 0
InconclusiveCount : 0
Time              : 00:01:21.0862719
TestResult        : {@{Describe=Establish Connectivity; Context=; Name=Connect to Rubrik Cluster; Result=Pas
                    ErrorRecord=; ParameterizedSuiteName=; Parameters=System.Collections.Specialized.Ordered
                    to vCenter Server; Result=Passed; Passed=True; Time=00:00:01.7606462; FailureMessage=; S
                    Parameters=System.Collections.Specialized.OrderedDictionary}, @{Describe=Create Live Mou
                    Passed=True; Time=00:00:02.5001648; FailureMessage=; StackTrace=; ErrorRecord=; Paramete
                    Parameters=System.Collections.Specialized.OrderedDictionary}, @{Describe=Create Live Mou
                    Result=Passed; Passed=True; Time=00:00:27.0919519; FailureMessage=; StackTrace=; ErrorRe
                    Parameters=System.Collections.Specialized.OrderedDictionary}...}
```

Output Results

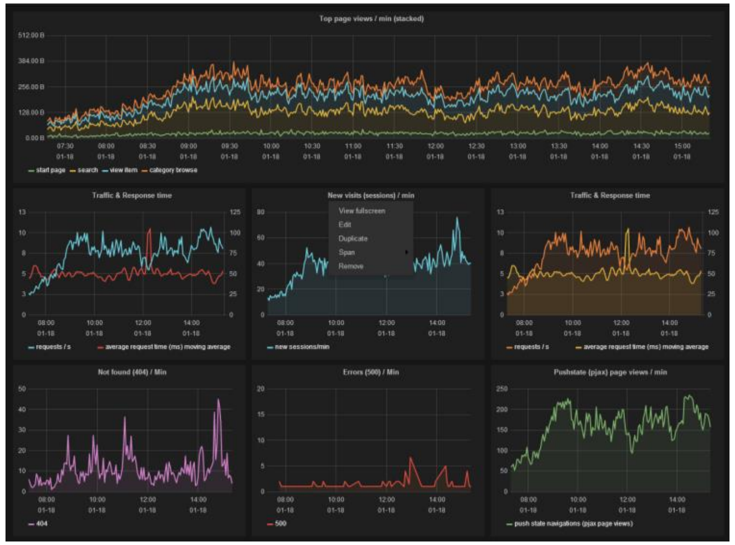


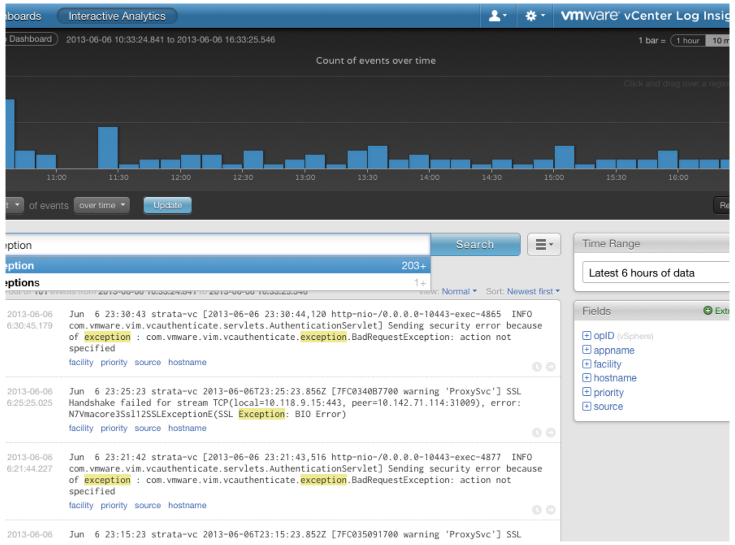
Help

Display 00 (Default)

Time	Priority	Hostname	Message
13:29:10	Cron.Alert	208.132.97.91	Sep 27 13:29:10 BJS-003-Win2003 SyslogGen Th
13:29:10	Local0.Critical	215.57.221.47	Sep 27 13:29:10 BJS-003-Win2003 SyslogGen Th
13:29:10	System3.Debug	222.96.144.194	Sep 27 13:29:10 BJS-003-Win2003 SyslogGen Th
13:29:10	Local5.Critical	199.148.120.158	Sep 27 13:29:10 BJS-003-Win2003 SyslogGen Th
13:29:10	Local6.Alert	212.225.146.101	Sep 27 13:29:10 BJS-003-Win2003 SyslogGen Th
13:29:10	System4.Alert	212.80.189.160	Sep 27 13:29:10 BJS-003-Win2003 SyslogGen Th
13:29:10	System2.Warning	224.82.223.225	Sep 27 13:29:10 BJS-003-Win2003 SyslogGen Th
13:29:10	System4.Notice	211.118.97.185	Sep 27 13:29:10 BJS-003-Win2003 SyslogGen Th
13:29:10	Local1.Warning	209.202.119.89	Sep 27 13:29:10 BJS-003-Win2003 SyslogGen Th
13:29:10	Lpr.Notice	220.201.229.221	Sep 27 13:29:10 BJS-003-Win2003 SyslogGen Th
13:29:10	Local5.Error	221.211.21.154	Sep 27 13:29:10 BJS-003-Win2003 SyslogGen Th
13:29:10	Local1.Notice	197.122.152.202	Sep 27 13:29:10 BJS-003-Win2003 SyslogGen Th
13:29:10	Local6.Alert	205.47.191.161	Sep 27 13:29:10 BJS-003-Win2003 SyslogGen Th
13:29:10	Mail.Info	221.19.157.147	Sep 27 13:29:10 BJS-003-Win2003 SyslogGen Th
13:29:10	Daemon.Error	208.238.142.66	Sep 27 13:29:10 BJS-003-Win2003 SyslogGen Th

100% 0 MPH





```
2013-06-06   Jun  6 23:30:43 strata-vc [2013-06-06 23:30:44,120 http-nio-/0.0.0.0-10443-exec-4865  INFO
6:30:45.179  com.vmware.vim.vcauthenticate.servlets.AuthenticationServlet] Sending security error because
             of exception : com.vmware.vim.vcauthenticate.exception.BadRequestException: action not
             specified
             facility  priority  source  hostname

2013-06-06   Jun  6 23:25:23 strata-vc 2013-06-06T23:25:23.856Z [7FC0340B7700 warning 'ProxySvc'] SSL
6:25:25.025  Handshake failed for stream TCP(local=10.118.9.15:443, peer=10.142.71.114:31009), error:
             N7Vmacore3Ssl12SSLExceptionE(SSL Exception: BIO Error)
             facility  priority  source  hostname

2013-06-06   Jun  6 23:21:42 strata-vc [2013-06-06 23:21:43,516 http-nio-/0.0.0.0-10443-exec-4877  INFO
6:21:44.227  com.vmware.vim.vcauthenticate.servlets.AuthenticationServlet] Sending security error because
             of exception : com.vmware.vim.vcauthenticate.exception.BadRequestException: action not
             specified
             facility  priority  source  hostname

2013-06-06   Jun  6 23:15:23 strata-vc 2013-06-06T23:15:23.852Z [7FC035091700 warning 'ProxySvc'] SSL
```

Syslog

Capture results for debugging and historical logging for the future

Grafana

Create a visual display of test runs to get a bird's eye view of all workflows

Other Tools

Smarter tools like VMware Log Insight can help pick apart trends and anomalies

The use cases for a **published**, **open**, and **documented** RESTful API are *endless*.

Questions?

vmworld

POSSIBLE BEGINS WITH YOU

vmware