Agile + DevOps WEST

A TECHWELL EVENT

AD22

DevOps Practices

3:00 PM

AD22 - The Lord of the Rings: DevOps Edition

Presented by:

Joseph Ours

Centric Consulting

Brought to you by:

TECHWELL

Joseph Ours

Joseph Ours has spent over two decades leading, coaching, speaking, and living the IT industry. Using his real-life, in-the-trenches experience, his view is radically different than most, a view focused on value individuals bring to an organization. He is the author of popular articles, such as, ‰ÛÏWhy You Need Women Testers‰Û , and speaker on behavior psychology topics in IT such as, ‰ÛÏThinking Fast and Slow for IT‰Û , Joseph has an ability to tackle tough topics and has sparked numerous penetrating and beneficial conversations that have been shared with folks from around the globe. Joseph serves as a National DevOps Practice Lead with the global firm, Centric Consulting. He has an MBA and is a Certified Project Management Professional where he continues to work with the best and brightest in the field.

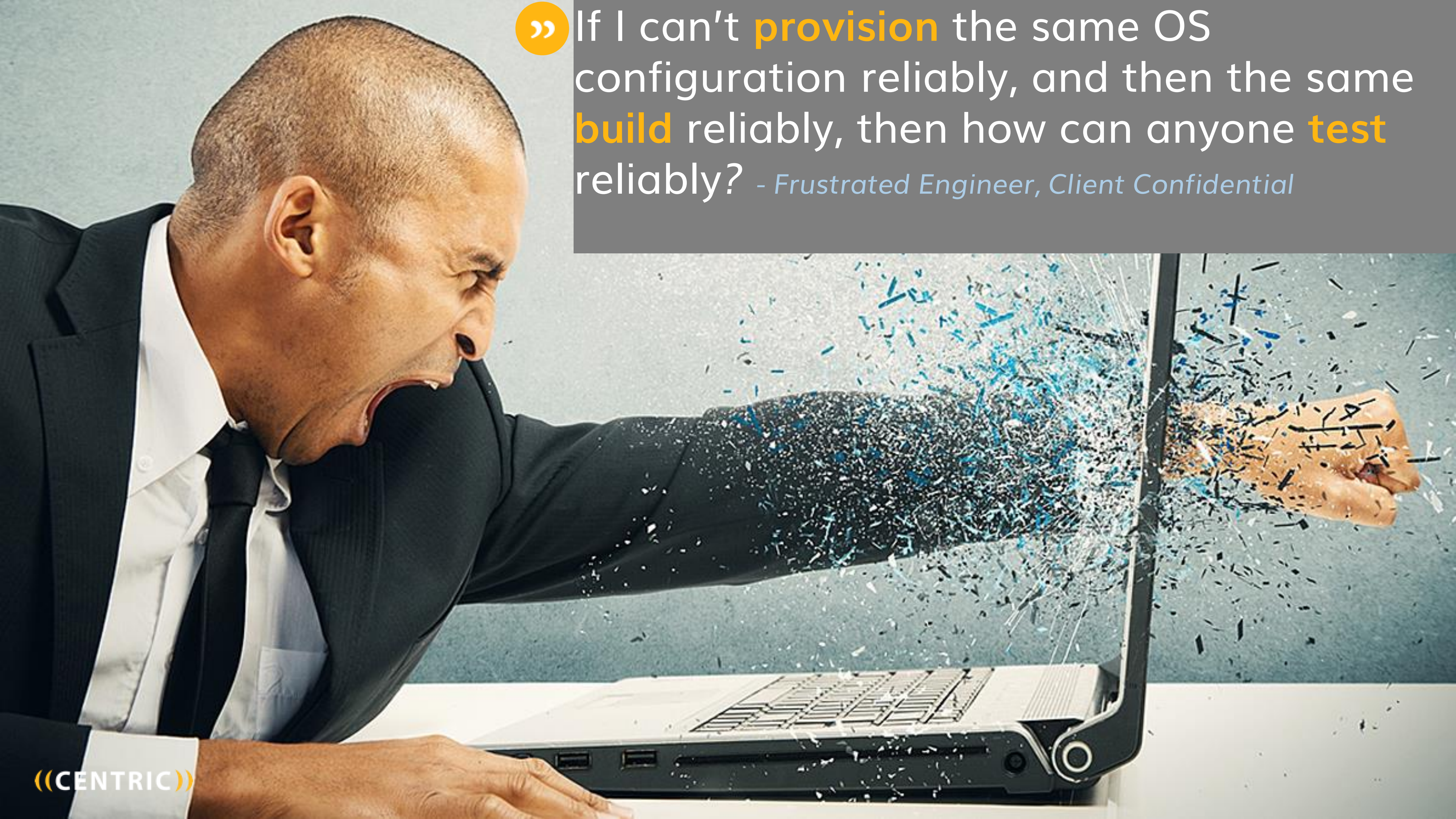
Cloud and Infrastructure Testing

Lessons Learned

((CENTRIC))

WHY?

((CENTRIC))



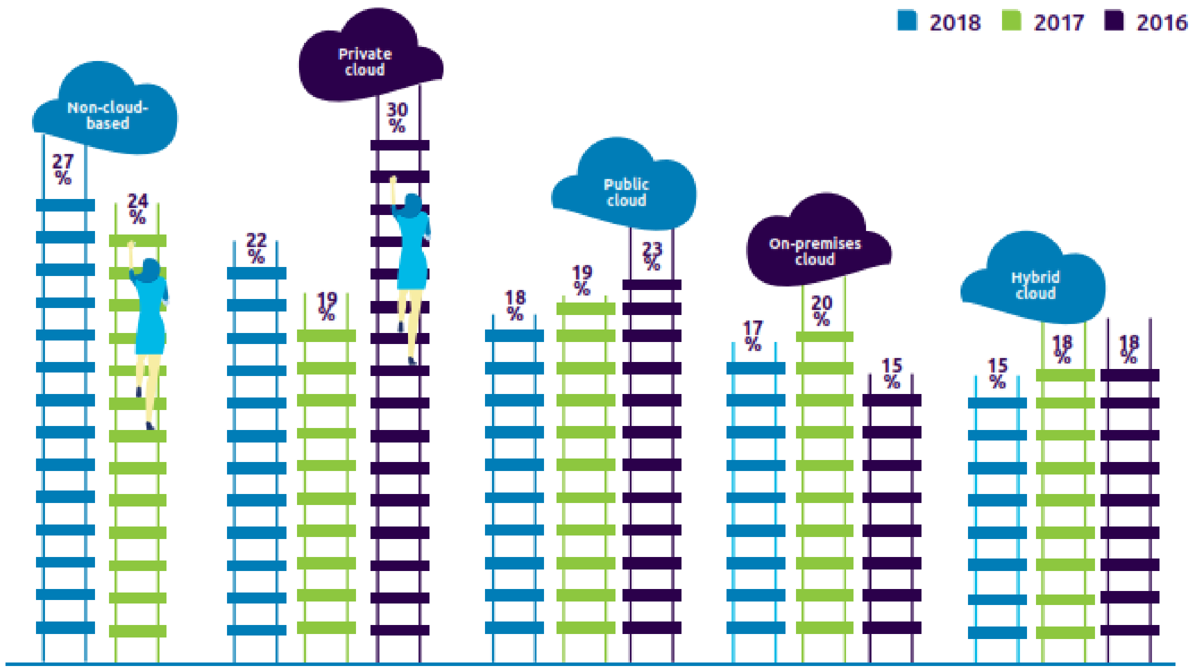
If I can't **provision** the same OS configuration reliably, and then the same **build** reliably, then how can anyone **test** reliably? *- Frustrated Engineer, Client Confidential*

((CENTRIC))

Cloud is Everywhere

According to the World Quality Report 2018-2019, 73% of all applications are hosted in some version of a cloud.

Figure 5

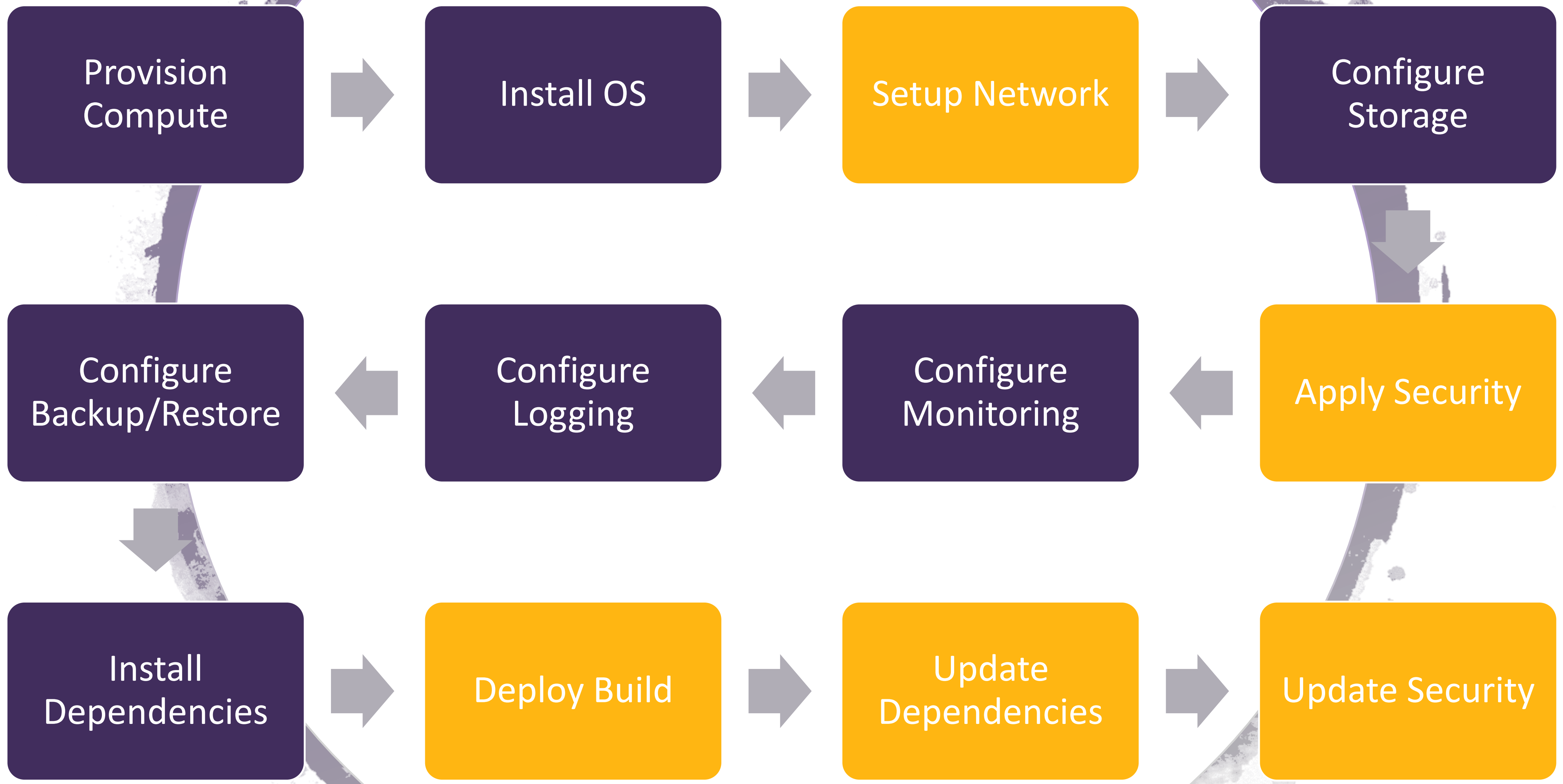


Speed Needs Speed

- As Agile is causing teams to develop faster, infrastructure needs to move faster as well.

- This has causes folks to start moving to Infrastructure as Code (IAC)

What Does IaC DO?



Simplified Example

Example – Environment Request

Simple Example

WebServer (Presentation Layer)

AppServer (Logic Layer)

DB Server (Data Layer)

Example – Environment Fulfillment

Network

- A segmented network with private and public subnets – some interconnectivity
- Entries into an Internet Gateway

Compute

- 2 machines
- OS installs (e.g. Your favorite approved Linux Kernel)

Security

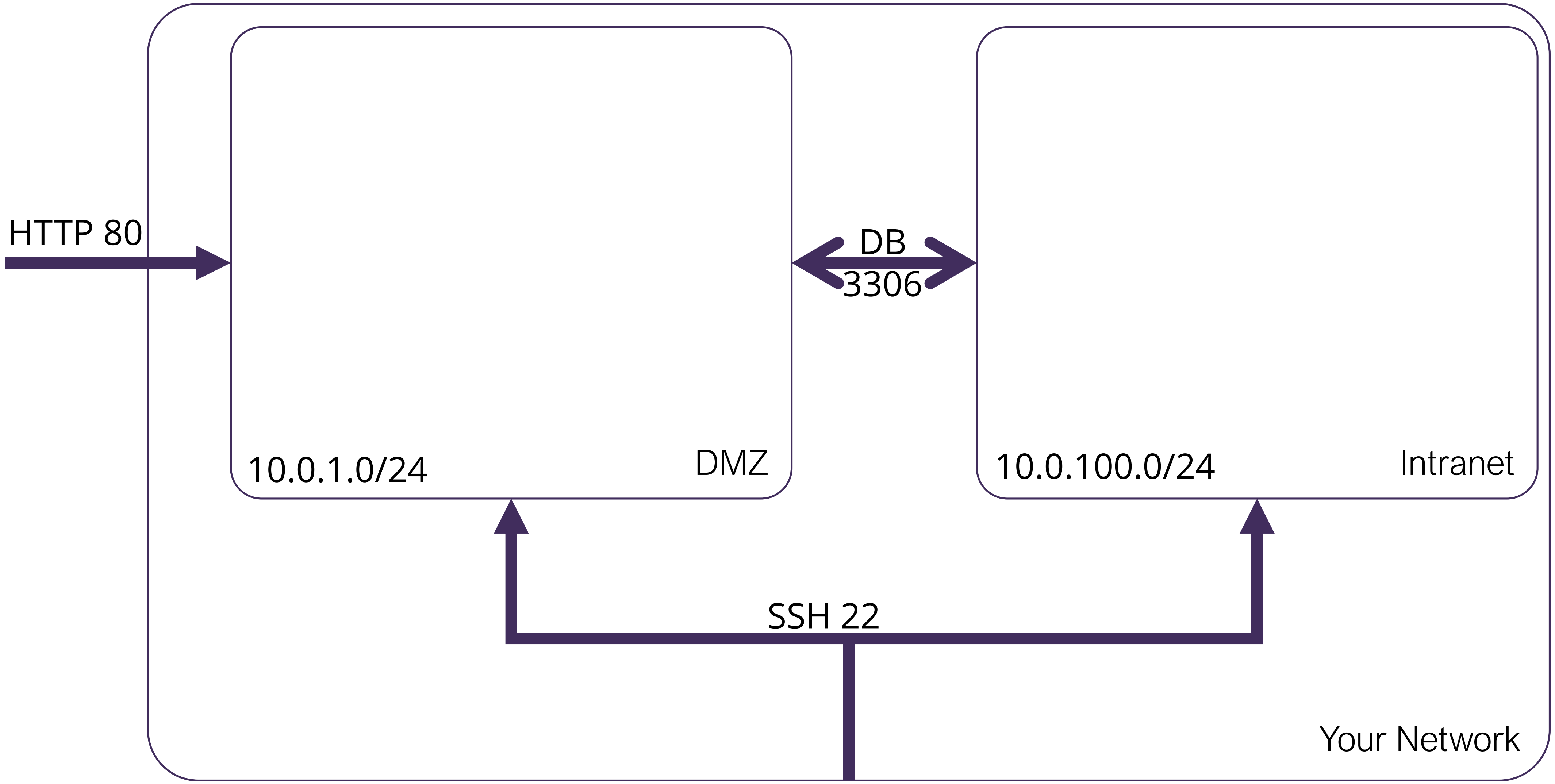
- Users and Roles
- Security Patches (e.g. What security needs to stop yelling at you)

Base applications

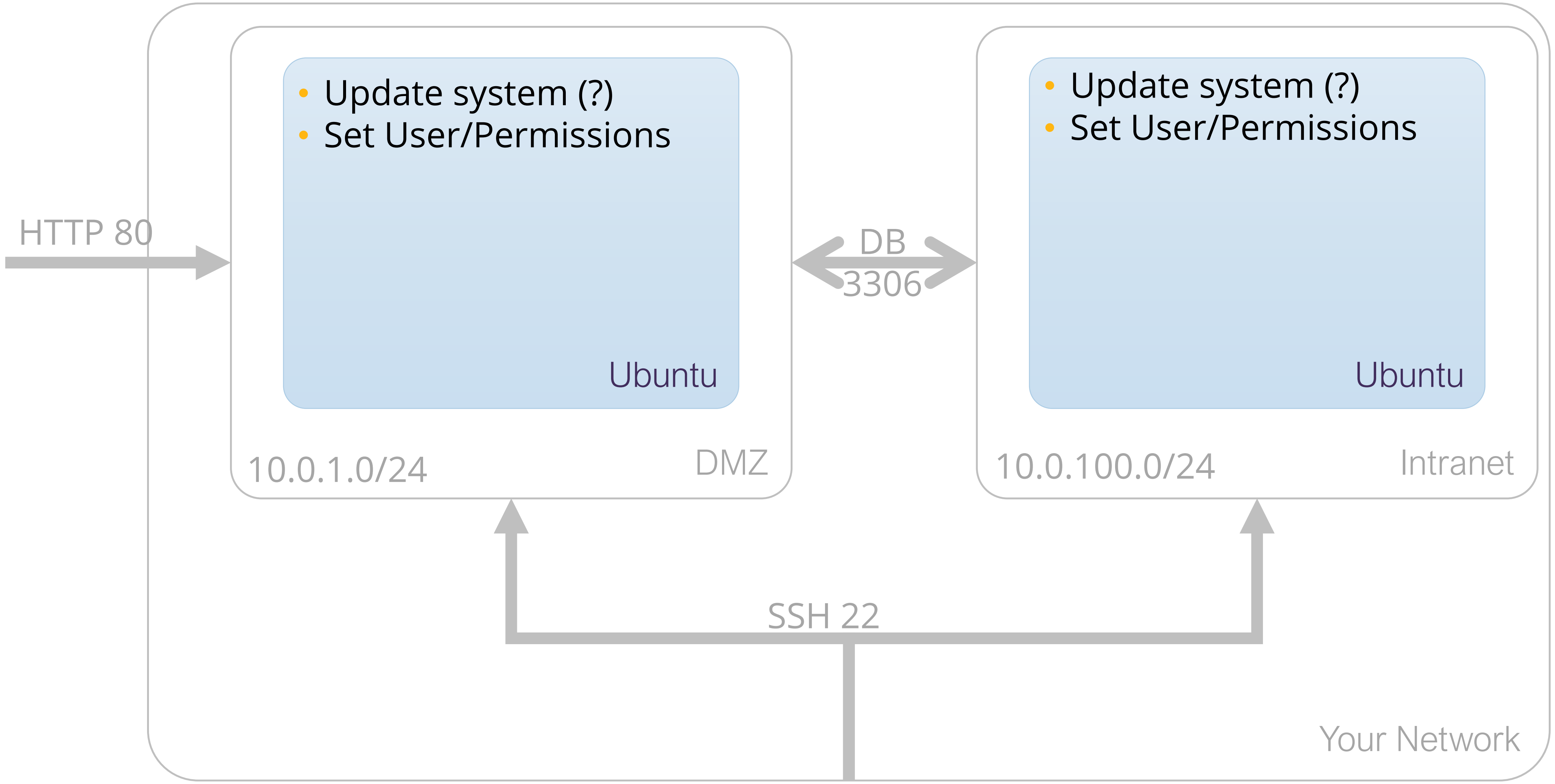
- e.g. Apache, MySQL

((CENTRIC))

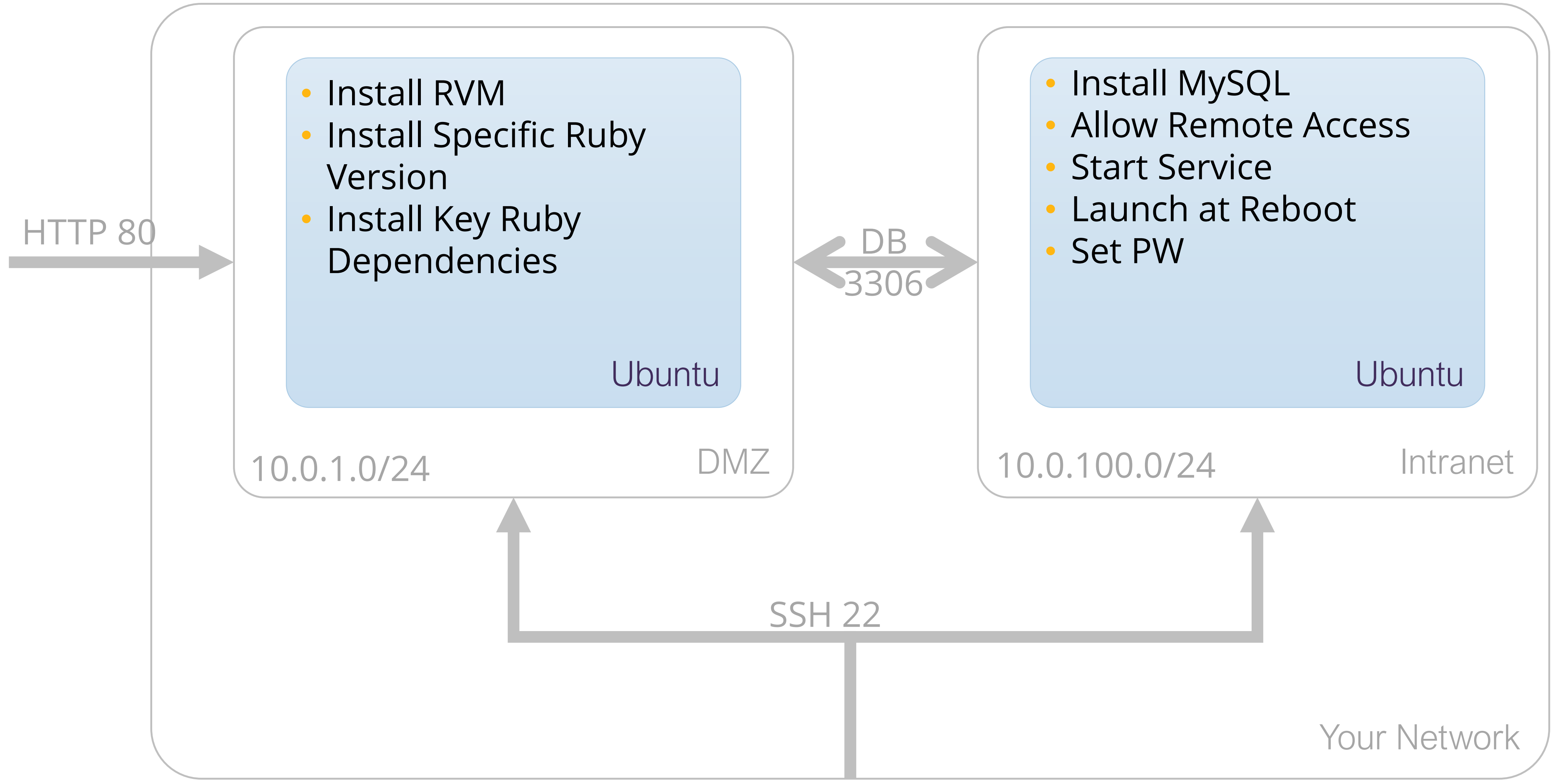
The Network


HTTP 80
DB
3306
10.0.1.0/24
DMZ
10.0.100.0/24
Intranet
SSH 22
Your Network

Operating System

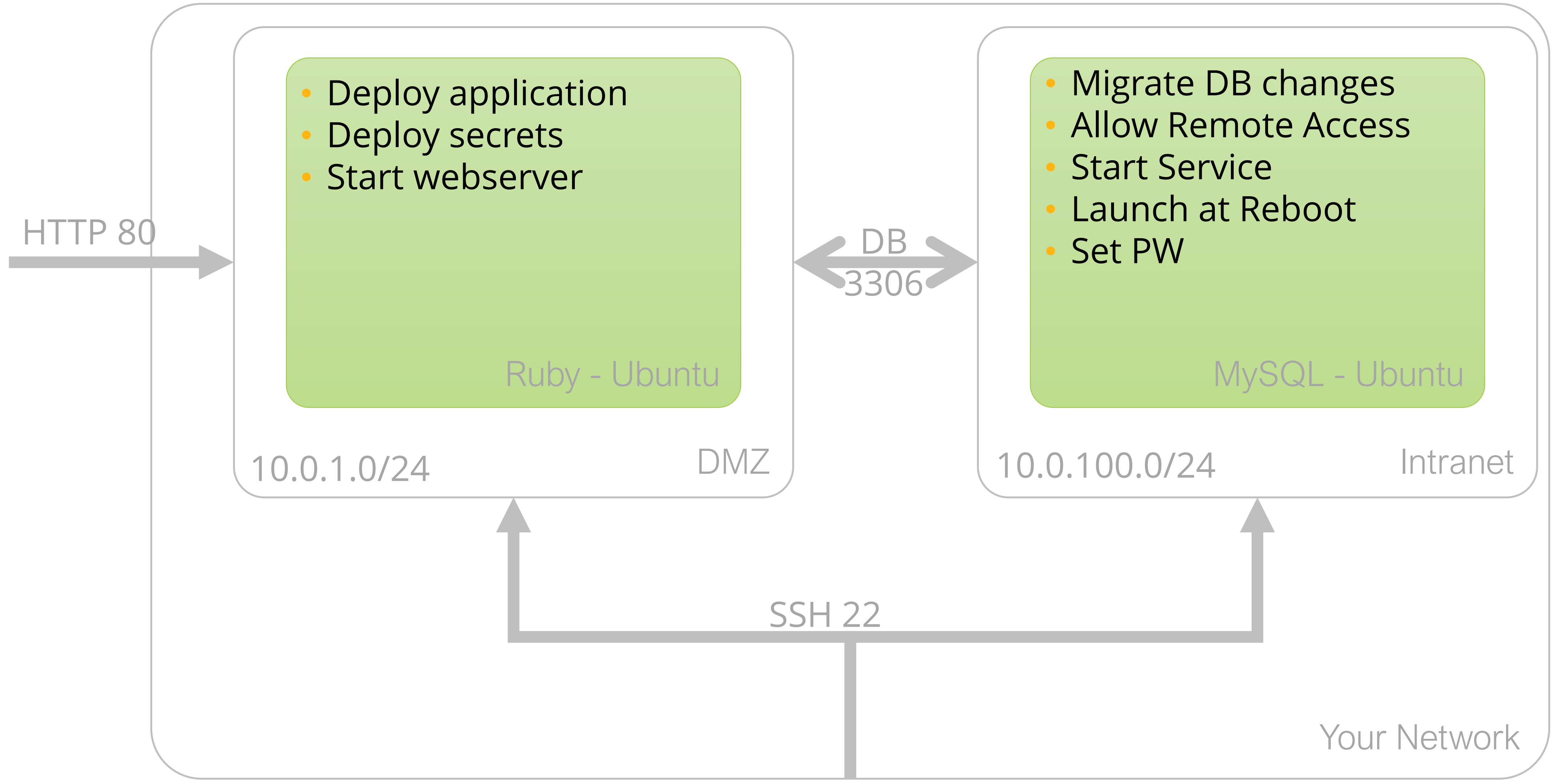


Base Applications

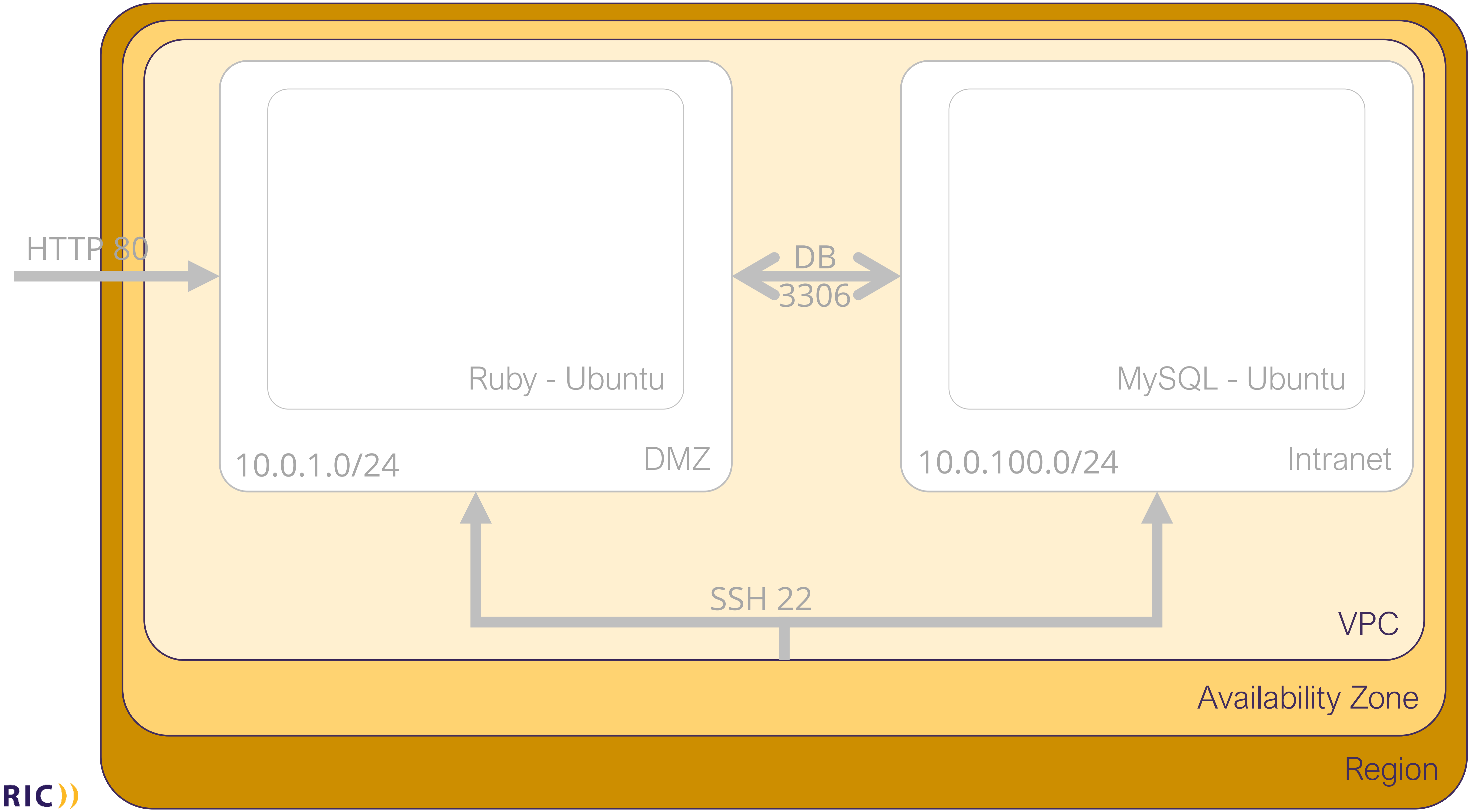

- Install RVM
- Install Specific Ruby Version
- Install Key Ruby Dependencies

- Install MySQL
- Allow Remote Access
- Start Service
- Launch at Reboot
- Set PW

Development Deployment Application

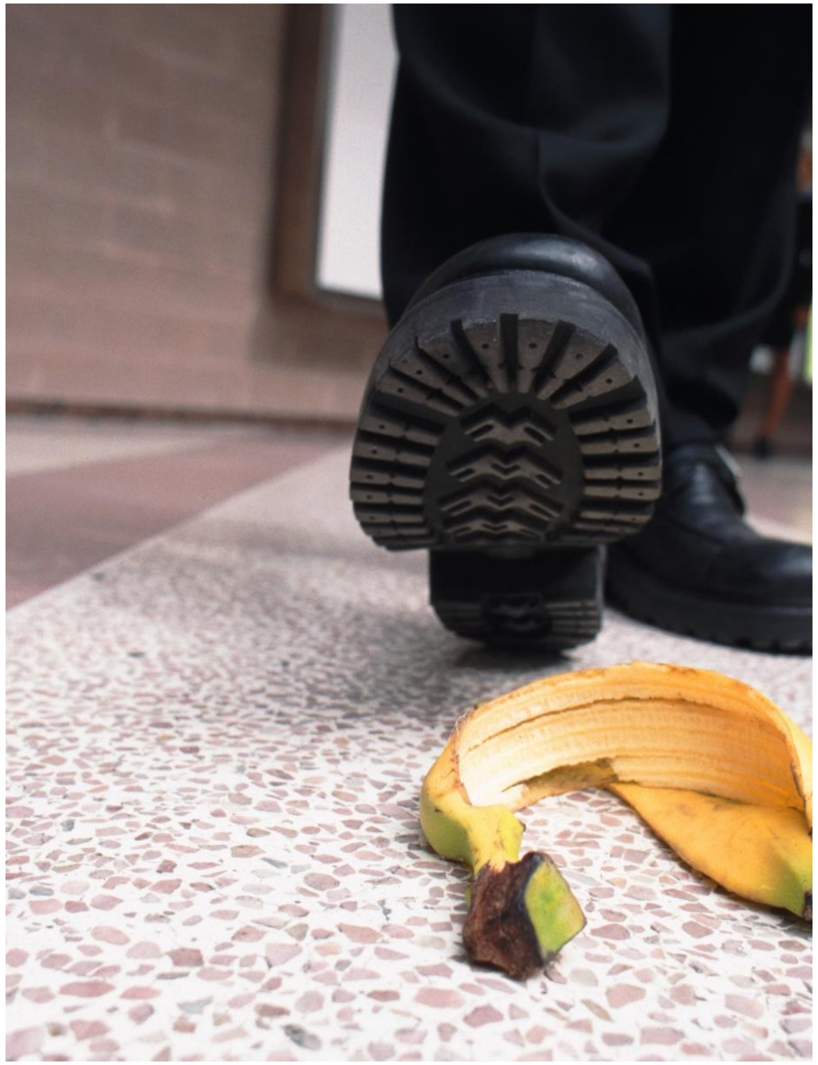


What If We Are in a Cloud?



((CENTRIC))

What Can Go Wrong?



- Port Assignment, Availability
- Credentials
- OS Configurations
- Base Application Configuration/Versioning
- Application Dependencies
- Cloud Regions, availability zones, and routing

Eliminate Human Error With IaC

- This is what led me down this path of test automation for infrastructure.
- Key Question: If we are going to use code, static or dynamic, to consistently establish infrastructure (and even the build/deploy process), shouldn't we test that it works the way we expect?

((CENTRIC))

This Is What IaC Looks Like

```
resource "aws_instance" "webserver" {
  ami                    = "${data.aws_ami.ubuntu.id}"
  instance_type          = "t2.micro"
  availability_zone      = "${var.aws_region}${var.aws_availability_zone}"
  vpc_security_group_ids = ["${aws_security_group.ssh.id}", "${aws_security_group.web.id}"]
  subnet_id              = "${aws_subnet.public.id}"
  key_name               = "MyEc2KeyPair"
  tags {
    Name = "webserver"
  }


  provisioner "remote-exec" {
    script = "bootstrap.sh"
    connection {
      type             = "ssh"
      user             = "ubuntu"
      private_key      = "${file("~/.ssh/MyEc2KeyPair.pem")}"
    }
  }
}
```

Where Can We Use Test Automation of IaC

NETWORK

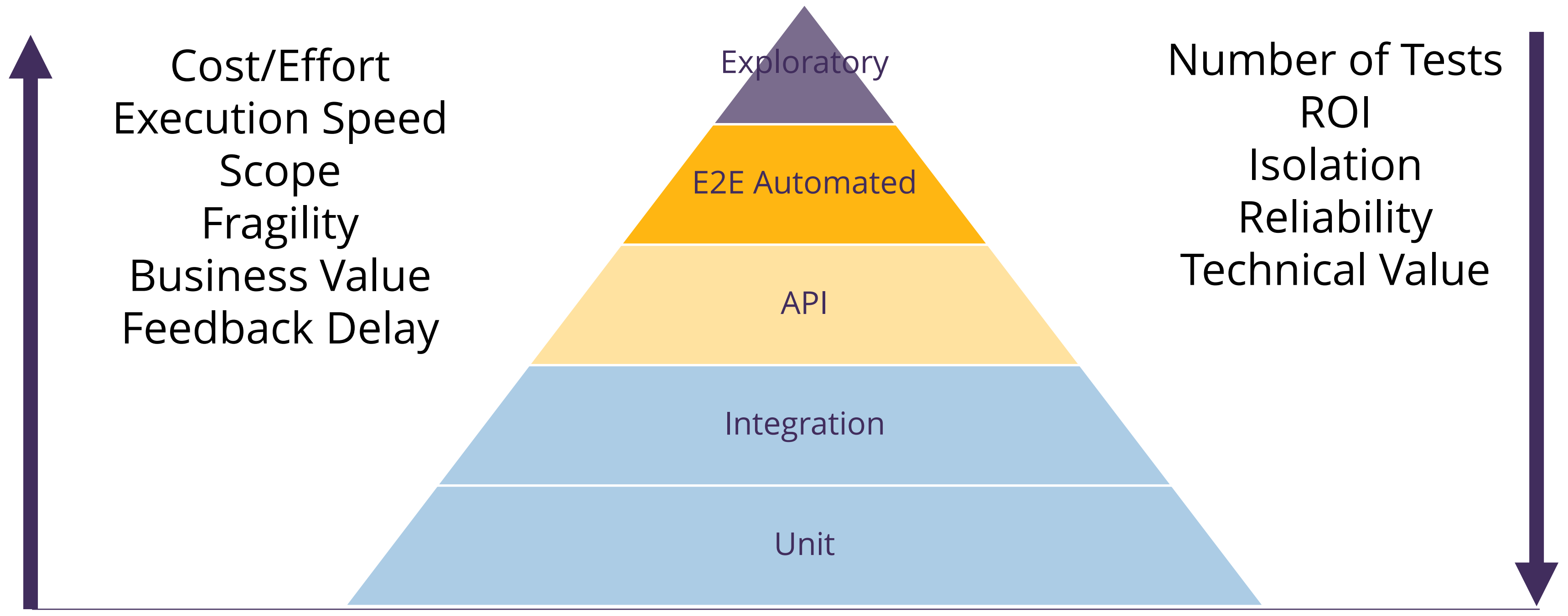
OS

BASE APPS

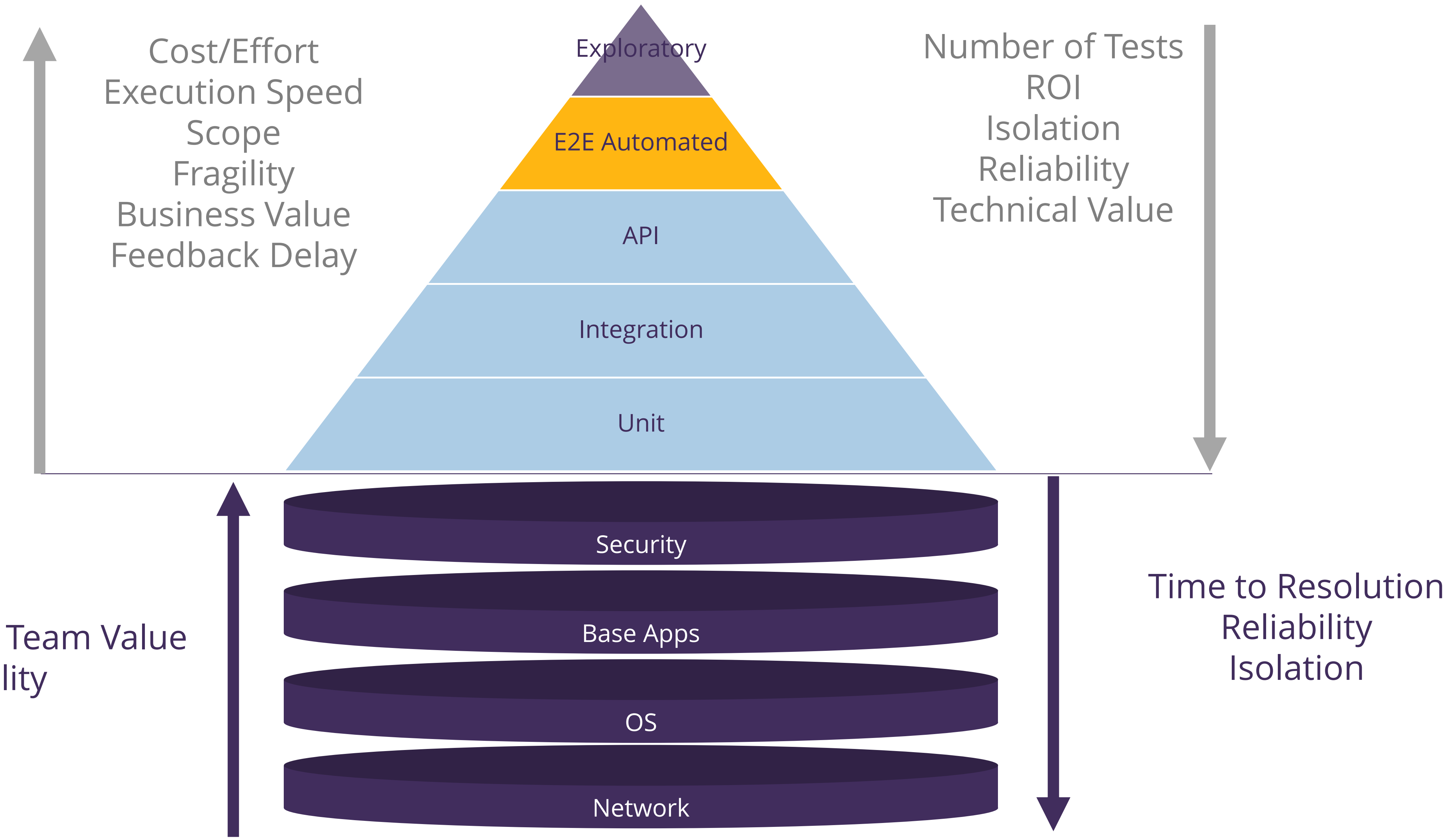
DEPLOYED APPS

SECURITY PRACTICES

Testing Pyramid Re-Visited



Testing Pyramid Re-Visited



Development Team Value
Fragility

((CENTRIC))

Primer

INSPEC

BY CHEF™

((CENTRIC))

This is NOT a Tutorial!

((CENTRIC))

What is Inspec

InSpec is an open-source testing framework for infrastructure with a human-readable language for specifying compliance, security and other policy requirements.

Runs anywhere against anything, including Docker, WinRM, etc…

Based on Ruby

From makers of Chef

```
describe file('/etc/myapp.conf') do
  it { should exist }
  its ('mode') { should cmp '0644' }
end

describe port(80) do
  it { should be_listening }
end
```

((CENTRIC))

Testing Cloud Configurations

```
describe aws_ec2_instance(name: 'webserver') do
  it { should be_running }
  its('image_id') { should eq 'ami-04169656fea786776' }
  its('instance_type') { should eq 't2.micro' }
  its('public_ip_address') { should eq '18.233.111.178' }
  its('vpc_id') { should eq 'vpc-04a315285bb4cce6e' }
  its('subnet_id') { should eq 'subnet-043b7e29765cebeeb' }
end
```

Testing Network Settings

```ruby
describe aws_vpc('vpc-04a315285bb4cce6e') do
  its('state') { should eq 'available' }
  its('cidr_block') { should eq '10.0.0.0/16' }
end

# private subnet
describe aws_subnet('subnet-0ac820f97f2cc0255') do
  it { should exist }
  its('vpc_id') { should eq 'vpc-04a315285bb4cce6e' }
  its('cidr_block') { should cmp '10.0.100.0/24' }
  its('availability_zone') { should eq 'us-east-1a' }
end

# public subnet
describe aws_subnet('subnet-043b7e29765cebeeb') do
  it { should exist }
  its('vpc_id') { should eq 'vpc-04a315285bb4cce6e' }
  its('cidr_block') { should cmp '10.0.1.0/24' }
  its('availability_zone') { should eq 'us-east-1a' }
end
```

Testing OS Settings

```
control 'linux' do
  impact 0.7.
  title 'Verify various linux settings'
  desc 'A few tests to show what can be done'

  describe port(22) do
    it { should be_listening }
  end

  describe port(8080) do
    it { should_not be_listening }
  end

  describe port(80) do
    it { should_not be_listening }
  end
end
```

((CENTRIC))

More OS Settings

```
describe file('/etc/shadow') do
  it { should exist }
  it { should be_file }
  it { should be_owned_by 'root' }
  its('group') { should eq 'shadow' }
  it { should_not be_executable }
  it { should_not be_readable.by('other') }
end

describe file('/etc/shadow') do
  it { should be_writable.by('owner') }
  it { should be_readable.by('owner') }
end

describe file('/etc/shadow') do
  it { should be_readable.by('group') }
end
```

Lots of OS Stuff to Test

```
control 'ssh-07' do
  impact 1.0
  title 'Client: Ask when checking host keys'
  desc "Don't automatically add new hosts keys to the list of known hosts."
  describe ssh_config do
    its('StrictHostKeyChecking') { should match(/ask|yes/) }
  end
end
```

((CENTRIC))

Demo

Let's See What This Looks Like

Complacency

- The Cloud makes it possible to do quite a bit magically.
- This magic often breeds complacency and/or apathy
- Here is what I learned on this endeavor

#1 – Testing Requires Thought

- Are we testing inherent functionality?
- Do all configurations need tested?
- Does the ability to configure need tested?
- What if there are no real environment requirements? What’s your source?
- Are there “new” sources to consider, e.g. Security

#2 – Understand What You're Testing and Your Tool

- Ensure you're testing the configuration, not the configuration file.

```
describe ssh_config do
  its('Protocol') { should eq("2")}
end
```

```
expect(ssh_content.find {|s| s.include?

"Protocol"}[/[\d]*\z/]).to eq("2")
```

- Understand your test target, how it works. In Linux many things are configuration files vs registry in Windows.

```
For instance, the ls command is provided by
the file /bin/ls, which holds the list of
machine instructions needed to display the
list of files in the current directory onto
the screen.
```

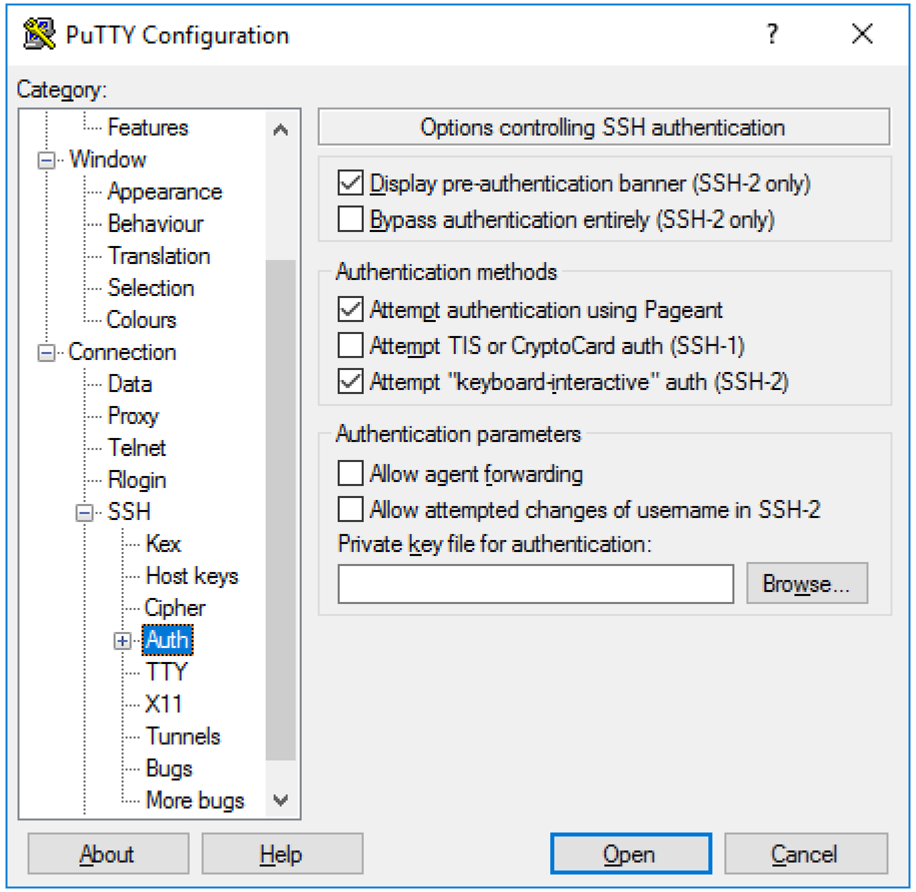
- In the cloud, size matter. Apps can fail due to memory constraints.

```
# Memory is TIGHT on a t2.micro so let's add some swap
space.
sudo dd if=/dev/zero of=/swapfile bs=512M count=8
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile
```

((CENTRIC))

#3 – SSH Is Your Frenemy, Ignorance Is Not

- SSH is how most infrastructure folks access their machines
- It's security, so....
- Get used to Command Line Interfaces (CLIs)
- Brush upon SSH options, they will either frustrate you and make your life easy



PuTTY Configuration

Category:

- Features
- Window
  - Appearance
  - Behaviour
  - Translation
  - Selection
  - Colours
- Connection
  - Data
  - Proxy
  - Telnet
  - Rlogin
  - SSH
    - Kex
    - Host keys
    - Cipher
    - Auth
    - TTY
    - X11
    - Tunnels
    - Bugs
    - More bugs

Options controlling SSH authentication

- [✓] Display pre-authentication banner (SSH-2 only)
- [ ] Bypass authentication entirely (SSH-2 only)

Authentication methods

- [✓] Attempt authentication using Pageant
- [ ] Attempt TIS or CryptoCard auth (SSH-1)
- [✓] Attempt "keyboard-interactive" auth (SSH-2)

Authentication parameters

- [ ] Allow agent forwarding
- [ ] Allow attempted changes of username in SSH-2

Private key file for authentication:

#3 – SSH Is Your Frenemy, Ignorance Is Not

```
Different Amazon Machine Images have different default users.  Most EC2
instances created with default
settings on AWS are Amazon Linux images.  This Demo uses an Ubuntu image.
Below are the Default user names for various AMI's
* For Amazon Linux 2 or the Amazon Linux AMI, the user name is ec2-user.
* For a Centos AMI, the user name is centos.
* For a Debian AMI, the user name is admin or root.
* For a Fedora AMI, the user name is ec2-user or fedora.
* For a RHEL AMI, the user name is ec2-user or root.
* For a SUSE AMI, the user name is ec2-user or root.
* For an Ubuntu AMI, the user name is ubuntu.
* Otherwise, if ec2-user and root don't work, check with the AMI provider.
```

AMI Instance names matter, even if they are all ubuntu version (ID’s)

#4 – Test Early, Test Often

- Test All the Things All the Time
    - Compute
    - Network
    - OS
    - Storage
    - Security
    - Cloud

- Test deployment configurations before your run regression tests
    - SSL Example

- Adapt to the concept of immutable environments, it’s a key DevOps capability

((CENTRIC))

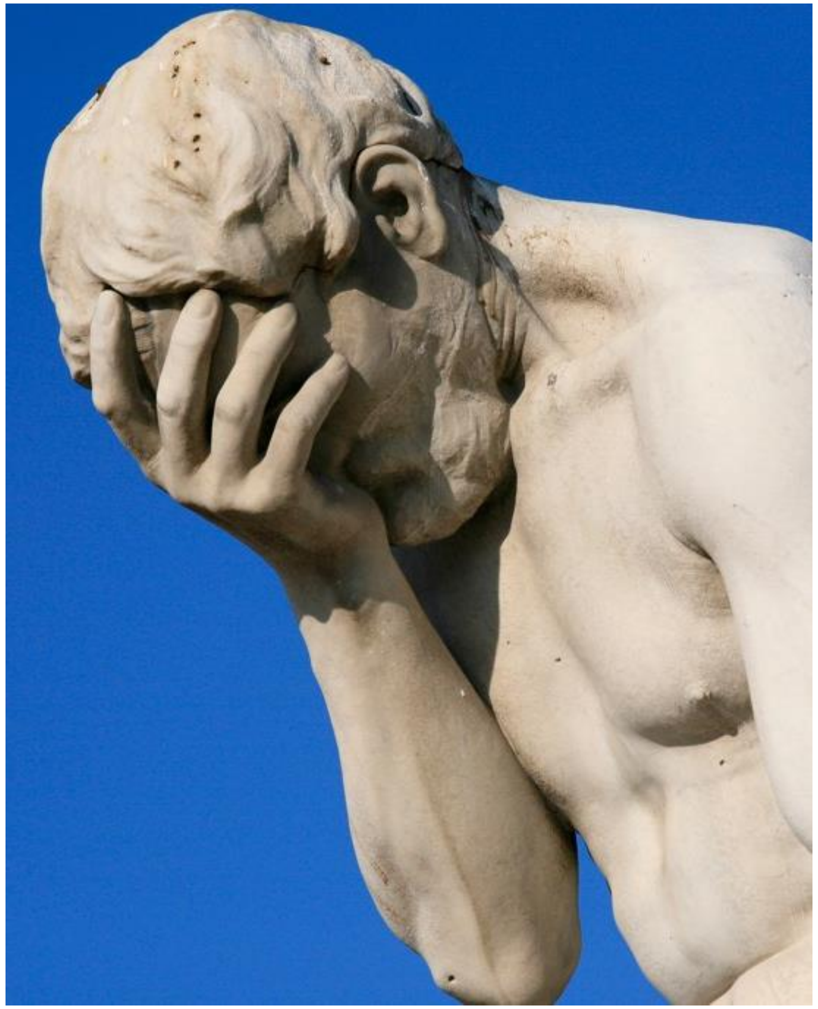
#5 – Continuously Learn

- Blogs
  - https://lollyrock.com/posts/inspec-terraform/
  - https://learn.chef.io/modules/inspec-aws-cloud
  - https://www.inspec.io/tutorials/

- (Some) Tools
  - https://www.inspec.io/
  - https://www.terraform.io
  - More Advanced
    - https://kitchen.ci
    - https://newcontext-oss.github.io/kitchen-terraform/tutorials/

Bonus – Pay Attention to Size and Orphaned Resources

- Ensure you spin up and spin down resources you need
- Allow for sufficient time - spinning up environments can take more time than you think, so can tearing them down
- I spent $800 in a month because I didn’t tear down unused resources



((CENTRIC))

Thank You

Joseph Ours

National Modern Software Delivery Practice Lead

joseph.ours@centricconsulting.com

CentricConsulting.com

614-668-2306

((CENTRIC))