RSAConference2018
San Francisco | April 16 – 20 | Moscone Center

SESSION ID: CSV-W12

EPHEMERAL DEVOPS: ADVENTURES IN MANAGING SHORT-LIVED SYSTEMS

Todd Carr

DevOps Engineer
Unity Technologies
@frozenfoxx

Who am I?

- DevOps Engineer at Unity Technologies
- Security Enthusiast
- Enormous fan of config management

Github:	frozenfoxx
Keybase:	frozenfoxx
Twitter:	@frozenfoxx

WHAT ARE EPHEMERAL SYSTEMS

What are Ephemeral Systems?

- Short-lived

What are Ephemeral Systems?

- Short-lived
- Light, middle, or heavyweight VMs

What are Ephemeral Systems?

- Short-lived
- Light, middle, or heavyweight VMs
- Dynamically deployed

What are Ephemeral Systems?

- Short-lived
- Light, middle, or heavyweight VMs
- Dynamically deployed
- Dynamically configured

What are Ephemeral Systems?

- Short-lived
- Light, middle, or heavyweight VMs
- Dynamically deployed
- Dynamically configured
- Dynamically destroyed

What are Ephemeral Systems?

- Short-lived
- Light, middle, or heavyweight VMs
- Dynamically deployed
- Dynamically configured
- Dynamically destroyed
- Usually heterogeneous

What did I build?

- Create and destroy about 600~1,000 heavyweight virtual machines an hour
  - Most of those run extremely CPU and disk intensive operations
- Updating existing and new VM configurations takes seconds
- Upgrades can be rolled out or rolled back in production extremely quickly
- Small team (three people) maintains it
- Bootstrapped with vSphere + Puppet

RSAConference2018

MATTERS NOW

WHY EPHEMERAL SYSTEMS?

Why Ephemeral Systems?

- Multiple immediately-available VMs

Why Ephemeral Systems?

- Multiple immediately-available VMs
- Non-containerized applications
  - Desktop apps
  - Legacy apps
  - Complex VMs

Why Ephemeral Systems?

- Multiple immediately-available VMs
- Non-containerized applications
  - Desktop apps
  - Legacy apps
  - Complex VMs
- Heterogeneous target pools
  - Multiple OSes
  - Multiple configurations
  - Multiple patch targets
  - Lots of iterative testing

Why Ephemeral Systems?

- Multiple immediately-available VMs
- Non-containerized applications
  - Desktop apps
  - Legacy apps
  - Complex VMs
- Heterogeneous target pools
  - Multiple OSes
  - Multiple configurations
  - Multiple patch targets
  - Lots of iterative testing
- Existing infrastructure
  - New flexibility without breaking anything
  - Doesn’t require buying new hardware

Why Ephemeral Systems?

- Testing
  - Rapid, immediate feedback with new code

Why Ephemeral Systems?

- Testing
  - Rapid, immediate feedback with new code
- Experimenting
  - Rapidly deploy on-the-fly changes

Why Ephemeral Systems?

- Testing
  - Rapid, immediate feedback with new code
- Experimenting
  - Rapidly deploy on-the-fly changes
- Simulating
  - Fully leverage dynamic environment configuration management tools
  - r10k (Puppet)
  - grinder (Salt)

Why Ephemeral Systems?

- Testing
  - Rapid, immediate feedback with new code
- Experimenting
  - Rapidly deploy on-the-fly changes
- Simulating
  - Fully leverage dynamic environment configuration management tools
  - r10k (Puppet)
  - grinder (Salt)
- Parallelization
  - Building
  - Testing

Why Ephemeral Systems?

- Testing
  - Rapid, immediate feedback with new code
- Experimenting
  - Rapidly deploy on-the-fly changes
- Simulating
  - Fully leverage dynamic environment configuration management tools
  - r10k (Puppet)
  - grinder (Salt)
- Parallelization
  - Building
  - Testing
- Don't have budget for new data centers or administrators

Why Ephemeral Systems in Security?

Exploit Development

- Write a revision, grab a target from multiple different pools of targets, destroy when done!
- Make a pool for every target
- Hook the grab, use, and destroy VM loop for every test script

Why Ephemeral Systems in Security?

Clean Slate Experimentation

- Rapidly deploy on-the-fly changes
- Simply call the API to destroy a machine at the conclusion of every test
- New machines for every run
- No more restore from snapshot

Why Ephemeral Systems in Security?

<u>Dynamic Behavior</u>

- Simulate changes in active installations
- Simply commit a change to a Hiera data file, run Puppet
- Need something even more dynamic? Make a Puppet Environment branch, deploy, and run the same machine against both branches
- No need to manually modify machines, all are still built from the same template

Why Ephemeral Systems in Security?

Narrowed Attack Window

- Non-containerized applications tend to stick around a long time
- Complex VM requirements
  - Non-Linux OSes
  - Specific patch levels
  - Custom software installations
- Treat these VMs as containers
  - Create, use, destroy, loop, all via API

Why Ephemeral Systems in Security?

Information Isolation

- No more wiping machines or rolling back to snapshots and hoping nothing is left on disk
- Grab a VM, use it, and dump it
- When the old one is destroyed it takes its environment with it, ensuring no disk recovery within the VM

TOOLS

Tools

- vSphere
  - VMs

Tools

- vSphere
  - VMs
- Puppet 4 (https://puppet.com/)
  - Agent, Server, PuppetDB
  - r10k

Tools

- vSphere
  - VMs
- Puppet 4 (https://puppet.com/)
  - Agent, Server, PuppetDB
  - r10k
- VmPooler (https://github.com/puppetlabs/vmpooler)

Tools

- vSphere
  - VMs, VM parameters
- Puppet 4 (https://puppet.com/)
  - Agent, Server, PuppetDB
  - r10k
- VmPooler (https://github.com/puppetlabs/vmpooler)
- Redis
- BIND
- ISC-DHCP-Server
- ***Dynamic DNS Updates from DHCP Server***
- rbvmomi

ISC Internet Systems Consortium

BIND
Berkeley Internet Name Domain

Tools

- vSphere
  - VMs, VM parameters
- Puppet 4 (https://puppet.com/)
  - Agent, Server, PuppetDB
  - r10k
- VmPooler (https://github.com/puppetlabs/vmpooler)
- Redis
- BIND
- ISC-DHCP-Server
- ***Dynamic DNS Updates from DHCP Server***
- rbvmomi
- Coffee

Internet Systems Consortium

BIND
Berkeley Internet Name Domain

BUILD

Build: Concepts

- Pools

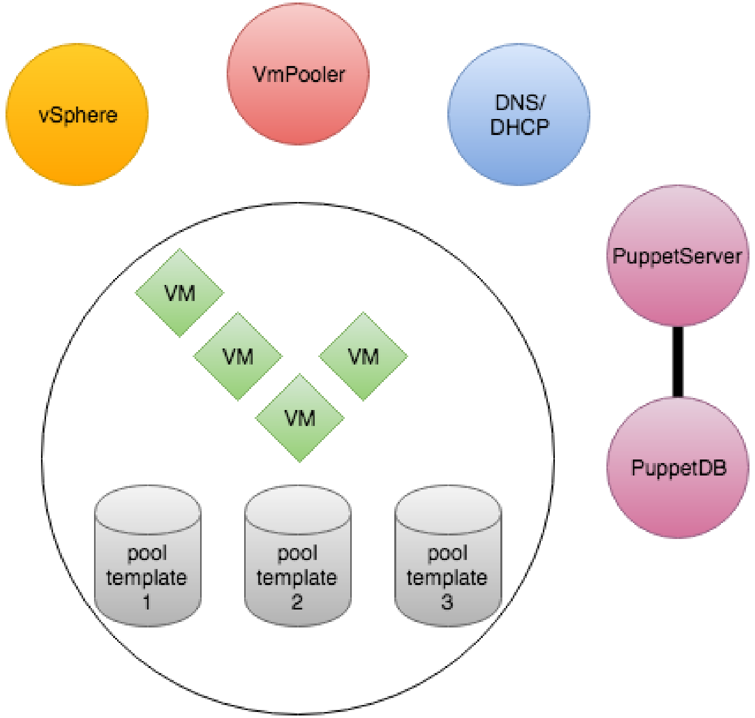
Build: Concepts

- Pools
- Self configuration
  - Puppet
  - Hiera
  - VMware GuestInfo Variables (hostname, pool, DNS, etc)

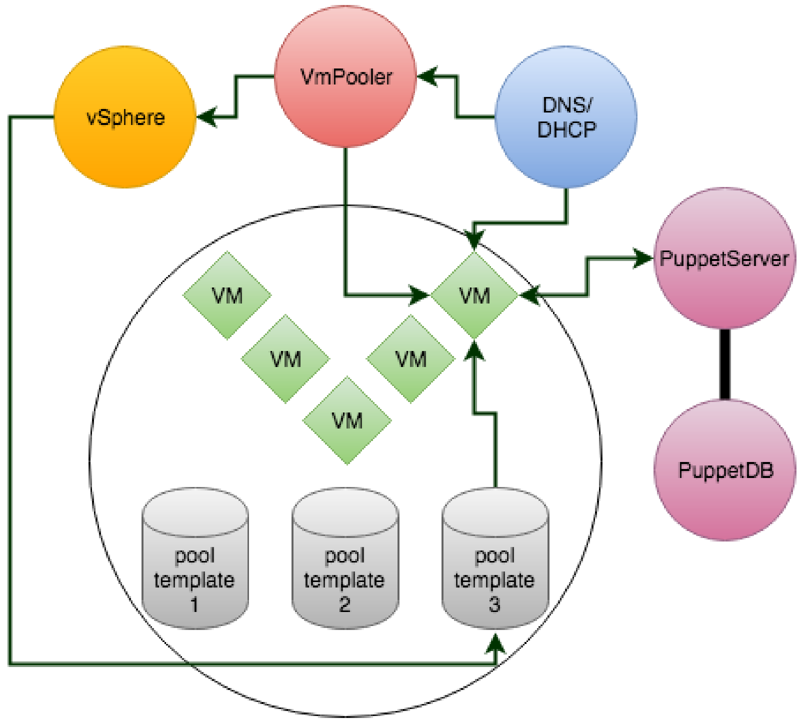
Build: Concepts

- Pools
- Self configuration
  - Puppet
  - Hiera
  - VMware GuestInfo Variables (hostname, pool, DNS, etc)
- Cleanup scripts

Build: Flow


vSphere
VmPooler
DNS/ DHCP
PuppetServer
PuppetDB
VM
VM
VM
VM
pool template 1
pool template 2
pool template 3

Build: Flow


vSphere
VmPooler
DNS/
DHCP
PuppetServer
PuppetDB
VM
VM
VM
VM
VM
pool
template
1
pool
template
2
pool
template
3

Build: Support

- Puppet
  - Autosigner (https://github.com/frozenfoxx/util/blob/master/puppet/puppet-autosign)
  - Certificate cleanup
    - Remove old & dead node certs, reinventory
    - https://github.com/frozenfoxx/util/blob/master/puppet/puppet-reap
  - Nodes cleaning script
    - Reports, facts, nodes
    - https://github.com/frozenfoxx/util/blob/master/puppet/puppet-cleanup-nodes

Build: Support

- Puppet
  - Autosigner (https://github.com/frozenfoxx/util/blob/master/puppet/puppet-autosign)
  - Certificate cleanup
    - Remove old & dead node certs, reinventory
    - https://github.com/frozenfoxx/util/blob/master/puppet/puppet-reap
  - Nodes cleaning script
    - Reports, facts, nodes
    - https://github.com/frozenfoxx/util/blob/master/puppet/puppet-cleanup-nodes
- VmPooler
  - Logrotate for vmpooler.log
  - Install provided init script

Build: Support

- Puppet
  - Autosigner (https://github.com/frozenfoxx/util/blob/master/puppet/puppet-autosign)
  - Certificate cleanup
    - Remove old & dead node certs, reinventory
    - https://github.com/frozenfoxx/util/blob/master/puppet/puppet-reap
  - Nodes cleaning script
    - Reports, facts, nodes
    - https://github.com/frozenfoxx/util/blob/master/puppet/puppet-cleanup-nodes
- VmPooler
  - Logrotate for vmpooler.log
  - Install provided init script
- vSphere
  - Ramdisk cleaner

Build: Monitoring

- Pools empty
- PuppetServer, PuppetDB down
  - Full disk
  - Too many files in a dir to remove
  - Certificates
- BIND/DHCP issues
  - Logging can get massive
- Weird vSphere things
  - Ramdisk fills up from creating/destroying VMs

RSAConference2018

MATTERS NOW

#RSAC

PERFORMANCE

Performance

- PuppetServer holds up well
  - 4 Cores, 16GB RAM, Linux
  - Around 600~1,000 VMs per hour
  - Load avg: 3.0 ~ 5.0
  - Creating certs, deleting certs, signing certs, compiling catalogs

Performance

- PuppetServer holds up well
  - 4 Cores, 16GB RAM, Linux
  - Around 600~1,000 VMs per hour
  - Load avg: 3.0 ~ 5.0
  - Creating certs, deleting certs, signing certs, compiling catalogs
- PuppetDB hold up extremely well
  - Not even phased by this usage, very low load

Performance

- PuppetServer holds up well
  - 4 Cores, 16GB RAM, Linux
  - Around 600~1,000 VMs per hour
  - Load avg: 3.0 ~ 5.0
  - Creating certs, deleting certs, signing certs, compiling catalogs
- PuppetDB hold up extremely well
  - Not even phased by this usage, very low load
- vSphere holds up okay
  - Linked clones are instantaneous (!)
  - vSphere VM itself may fall over, taking the API with it
  - Needs restarting every six to nine months, YMMV

Performance

- PuppetServer holds up well
  - 4 Cores, 16GB RAM, Linux
  - Around 600~1,000 VMs per hour
  - Load avg: 3.0 ~ 5.0
  - Creating certs, deleting certs, signing certs, compiling catalogs
- PuppetDB hold up extremely well
  - Not even phased by this usage, very low load
- vSphere holds up okay
  - Linked clones are instantaneous (!)
  - vSphere VM itself may fall over, taking the API with it
  - Needs restarting every six to nine months, YMMV
- DHCP/BIND holds up okay...mostly
  - Once a year or so stops adding/removing, just restart

USAGE

Usage: General

- Get a box
  - *curl -d --url vmpooler.somewhere.com:4567/api/v1/vm/[vm-type]*
  - Checks out a box, [box hostname]
- Use that box
- All done? Dump the box
  - *curl -X DELETE --url vmpooler.somewhere.com:4567/api/v1/vm/[box hostname]*
- Loop

Usage: Parallel Testing Batches

- Array of tests
- Get boxes
  - Loop over retrieval for array of boxes
  - *curl -d --url vmpooler.somewhere.com:4567/api/v1/vm/[vm-type]*
- Run block of tests against array of boxes
- All done? Dump the boxes
  - Loop over array of boxes
  - *curl -X DELETE --url vmpooler.somewhere.com:4567/api/v1/vm/[box hostname]*
- Loop

Usage: Dynamic Environments

- New Puppet branch, need to test
- Get a box
  - *curl -d --url vmpooler.somewhere.com:4567/api/v1/vm/[vm-type]*
  - Checks out a box, [box hostname]
- SSH to that box
- Let's config that box
  - Normal mode: *puppet agent --test*
  - New feature: *puppet agent --test --environment [featurebranch]*
- All done? Dump the box
  - *curl -X DELETE --url vmpooler.somewhere.com:4567/api/v1/vm/[box hostname]*
- Loop
- Merge Puppet branch

Usage: Dynamic App Behavior

- Make a new Puppet environment, *[newbehavior]*
  - Users, configs, whatever needs to be simulated in Hiera and Manifests
  - Deploy with *r10k*
- Get a box
  - *curl -d --url vmpooler.somewhere.com:4567/api/v1/vm/[vm-type]*
- SSH to that box, alter the app behavior
  - Normal behavior: *puppet agent –test*
  - New behavior: *puppet agent --test --environment [newbehavior]*
- Test
- All done? Dump the box and branch
  - *curl -X DELETE --url vmpooler.somewhere.com:4567/api/v1/vm/[box hostname]*
  - *git push origin :[newbehavior]*

MAINTENANCE

Maintenance

- These examples are using Puppet
- These sorts of concerns will affect ANY tool doing config management
  - Salt, Chef, CFengine, Puppet, all have the same concerns
  - They all expect nodes to live a long time
- Maintenance is...different
- Ephemeral VMs die all the time, that’s okay
- If any component dies, the pools drain
- Drained pools are bad
- Bad pools are sad pools

Maintenance

- vmpooler.log
  - Size, rotation, needs monitoring

Maintenance

- vmpooler.log
  - Size, rotation, needs monitoring
- Puppet Cleanup

Maintenance

- vmpooler.log
  - Size, rotation, needs monitoring
- Puppet Cleanup
  - Certificate allocation

Maintenance

- vmpooler.log
  - Size, rotation, needs monitoring
- Puppet Cleanup
  - Certificate allocation
  - Certificate cleanup

Maintenance

- vmpooler.log
  - Size, rotation, needs monitoring
- Puppet Cleanup
  - Certificate allocation
  - Certificate cleanup
  - Certificate autosigning

Maintenance

- vmpooler.log
  - Size, rotation, needs monitoring
- Puppet Cleanup
  - Certificate allocation
  - Certificate cleanup
  - Certificate autosigning
  - PuppetServer JVM/CPU allocation

Maintenance

- vmpooler.log
  - Size, rotation, needs monitoring
- Puppet Cleanup
  - Certificate allocation
  - Certificate cleanup
  - Certificate autosigning
  - PuppetServer JVM/CPU allocation
  - PuppetDB JVM/CPU allocation

Maintenance

- vmpooler.log
  - Size, rotation, needs monitoring
- Puppet Cleanup
  - Certificate allocation
  - Certificate cleanup
  - Certificate autosigning
  - PuppetServer JVM/CPU allocation
  - PuppetDB JVM/CPU allocation
  - PuppetDB database upgrades/migration

Maintenance

- vmpooler.log
  - Size, rotation, needs monitoring
- Puppet Cleanup
  - Certificate allocation
  - Certificate cleanup
  - Certificate autosigning
  - PuppetServer JVM/CPU allocation
  - PuppetDB JVM/CPU allocation
  - PuppetDB database upgrades/migration
  - Losing PuppetServer/PuppetDB certificates (means rebuild)

Maintenance

- vmpooler.log
  - Size, rotation, needs monitoring
- Puppet Cleanup
  - Certificate allocation
  - Certificate cleanup
  - Certificate autosigning
  - PuppetServer JVM/CPU allocation
  - PuppetDB JVM/CPU allocation
  - PuppetDB database upgrades/migration
  - Losing PuppetServer/PuppetDB certificates (means rebuild)
  - Disk filling up

Maintenance

- vmpooler.log
  - Size, rotation, needs monitoring
- Puppet Cleanup
  - Certificate allocation
  - Certificate cleanup
  - Certificate autosigning
  - PuppetServer JVM/CPU allocation
  - PuppetDB JVM/CPU allocation
  - PuppetDB database upgrades/migration
  - Losing PuppetServer/PuppetDB certificates (means rebuild)
  - Disk filling up
  - Failing to self-cleanup (too many files)

Maintenance

- vmpooler.log
  - Size, rotation, needs monitoring
- Puppet Cleanup
  - Certificate allocation
  - Certificate cleanup
  - Certificate autosigning
  - PuppetServer JVM/CPU allocation
  - PuppetDB JVM/CPU allocation
  - PuppetDB database upgrades/migration
  - Losing PuppetServer/PuppetDB certificates (means rebuild)
  - Disk filling up
  - Failing to self-cleanup (too many files)
- vSphere stops responding

EXTENDING

Extending

- TerraForm + Packer
  - TerraForm for management hosts
  - Packer for management hosts & Ephemeral VMs

Extending

- TerraForm + Packer
  - TerraForm for management hosts
  - Packer for management hosts & Ephemeral VMs
- ChatOps
  - Calls to VmPooler API

Extending

- TerraForm + Packer
  - TerraForm for management hosts
  - Packer for management hosts & Ephemeral VMs
- ChatOps
  - Calls to VmPooler API
- Containers for management components
  - VmPooler has a container, but it includes Redis (heavy)
  - Redis containers exist
  - Puppet containers aren’t 100% supported (but work!)

Extending

- TerraForm + Packer
  - TerraForm for management hosts
  - Packer for management hosts&Ephemeral VMs
- ChatOps
  - Calls to VmPooler API
- Containers for management components
  - VmPooler has a container, but it includes Redis (heavy)
  - Redis containers exist
  - Puppet containers aren't 100% supported (but work!)
- Removing PuppetDB
  - If you aren't using the data or collections, it can only fail here
  - Lose speed on compilation, YMMV

Extending

- One more wild idea

Extending

- One more wild idea
- Remove PuppetServer from Ephemeral VM loop
  - Go full standalone
  - Use“puppet apply”to self-configure
  - Use Packer scripts to prebuild only parts from Hiera and Codebase relevant to an Ephemeral VM type
  - Lose flexibility for testing quickly
  - Gain reliability on the server side
  - No more certificate cleanup

SUMMARY

Summary

- VmPooler is awesome!

Summary

- VmPooler is awesome!
- Dynamic Environments are awesome!

Summary

- VmPooler is awesome!
- Dynamic Environments are awesome!
- Puppet is awesome!

Summary

- VmPooler is awesome!
- Dynamic Environments are awesome!
- Puppet is awesome!
- Dynamic DNS + DHCP is awesome!

Summary

- VmPooler is awesome!
- Dynamic Environments are awesome!
- Puppet is awesome!
- Dynamic DNS + DHCP is awesome!
- Dynamic pools are awesome!

Summary

- VmPooler is awesome!
- Dynamic Environments are awesome!
- Puppet is awesome!
- Dynamic DNS + DHCP is awesome!
- Dynamic pools are awesome!
- Everything is awesome!

Apply What You Have Learned Today

- Deploy toolchain VMs
  - Vmpooler, DHCP + Bind, PuppetServer + PuppetDB
- Reconfigure BIND for Dynamic DNS Updates
- Create pool templates
  - OSes, patch levels, installed software, desired targets
- Experiment!
  - Exploit Development (*Usage: General | Parallel Testing Batches*)
  - Clean Slate Experimentation (*Usage: General | Dynamic Environment*)
  - Dynamic Behavior (*Usage: Dynamic App Behavior*)
  - Narrowed Attack Windows (*Usage: General | Dynamic App Behavior*)
  - Information Isolation (*Usage: General | Dynamic Environment*)

QUESTIONS