WHITE PAPER – **AUGUST 2017**

AN INTRODUCTION TO BOSH

by VMware

vmware®

Table of Contents

What is BOSH?

BOSH Overview

BOSH is an open source tool that enables deployment and lifecycle management of distributed systems. It is the primary method used to deploy Cloud Foundry and is contributed to by many key members of the Cloud Foundry Foundation, such as Google, Pivotal, & VMware. It can support deployments across many different IaaS providers. Some of these providers are:

- VMware vSphere
- Google Compute Platform
- Amazon Web Services EC2
- Microsoft Azure
- OpenStack

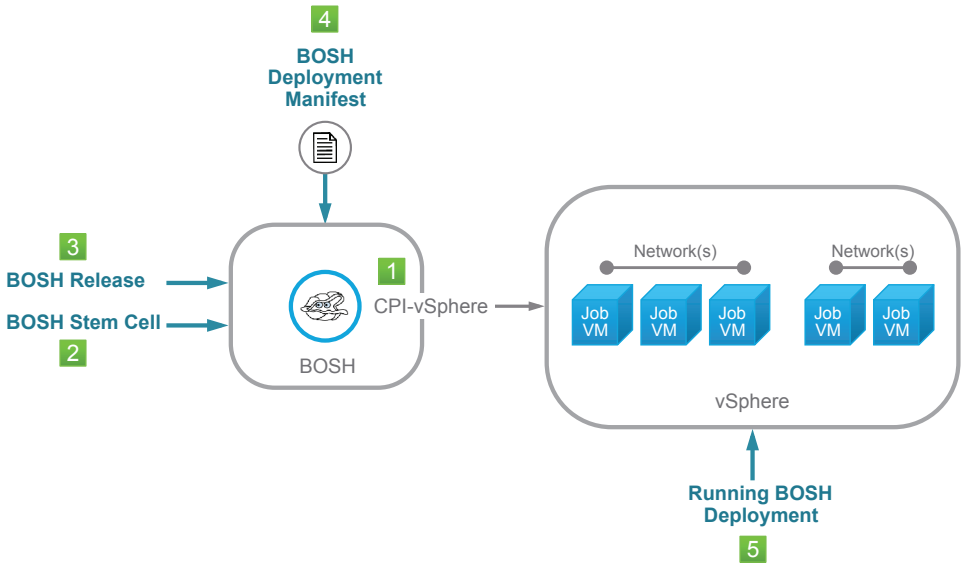

Figure 1: BOSH Overview

BOSH accomplishes deployments by creating several major abstraction objects that make it easy and repeatable to deploy complex systems. Referencing the figure above, these objects include:

1. **CPI:** The cloud provider interface, or CPI, is the executable library BOSH uses to interact with any given IaaS. One CPI is available for every BOSH-supported IaaS, and when you deploy the BOSH instance(s) you can define which one(s) it will use. In the image above, a vSphere CPI is shown. It allows BOSH to perform all the required IaaS actions, such as creating a VM or instance, as well as various other instance, network, and storage primitives required to instantiate a deployment.

2. **BOSH stemcell:** A stemcell is a versioned base operating system image built for each CPI that BOSH supports. It is commonly based on Canonical's Ubuntu distribution, but is also available in RHEL and even Windows image ports. Typically, the stemcell is a hardened base OS image with a BOSH agent pre-deployed. BOSH will use this agent to install and manage the lifecycle of software on that VM for instance.
3. **BOSH release:** A BOSH release is a versioned tarball containing the complete source code and job definitions required to describe to BOSH how that release of software should be deployed on a VM or instance provisioned from a stemcell. An example is the Kubo release which includes all the packages and details required to allow BOSH deploy a fully functional Kubernetes cluster.
4. **BOSH deployment manifest:** BOSH needs to receive some declarative information to actually deploy something. This is provided by an operator via a manifest. A manifest defines one or more *releases* and *stemcells* to be used in a deployment and provides some key variables like IPstack info, instance count, and advanced configuration of the given release(s) you want to deploy. This manifest is typically written in a YAML format.
5. **BOSH deployment:** BOSH needs some declarative information before it can deploy anything. This is provided by an operator via a deployment manifest and a cloud-config manifest. These manifests are typically written in a YAML format.
   - *cloud-config manifest:* This YAML is specific to an IaaS as defined by the properties made available in its CPI. It will provide definitions for things like networks, VM sizes, storage locations, and availability zone mappings. This manifest is global, which means that there can be only one instance per BOSH, and can be referred to by multiple deployment manifests.
   - *deployment-manifest:* This manifest refers to objects in the cloud-config and focuses on properties for the releases. The manifests define one or more releases and stemcells to be used in a deployment and provide some key variables like instance count and advanced configuration of the given release(s) to be deployed. This allows for deployment manifests to be portable across CPIs.

What Problems Does BOSH Solve?

BOSH lets release developers[1] easily version, package, and deploy software in a reproducible manner. Operators can consume BOSH releases and be guaranteed that deployments are repeatable with predictable results across environments. To accomplish this, BOSH lets release developers focus on providing some key abilities when building a release:

- **Identifiability**

An operator needs to be able to document the deployment of software and its versions. A BOSH release, by design, requires the developer to declare and package everything in the release. The release itself must also be versioned. This allows an operator to fully understand what is deployed as well as consistently upgrade or downgrade versions of software in a release.

[1] A release developer is a developer who packages software to be deployed by BOSH.

Example: In Figure 2, an operator defining a deployment can refer to one or more versioned releases in a deployment manifest. This allows for identification of the software versions used. In the image above, BOSH has two versions of the Kubo release available: versions 0.0.5 and 0.0.6. The operator has defined the use of version 0.0.5 of the release in the deployment manifest, which will enforce the use of Kubernetes version 1.6.6 across the deployment called "mykubo-deployment."

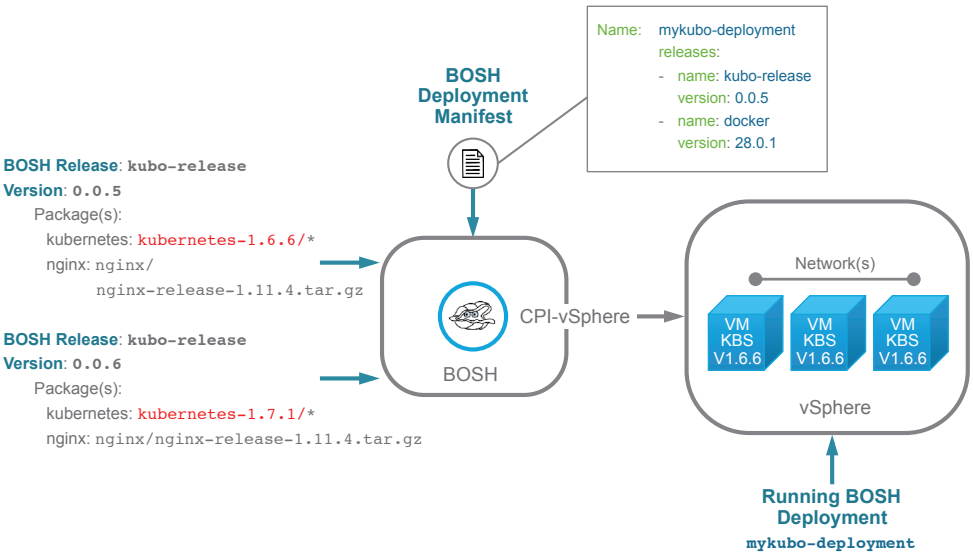

Figure 2: BOSH Identifiability

* Reproducibility

Another key tenant in releasing software addressed by BOSH is reproducibility. To an operator, this means that software should be deployed in exactly the same way across multiple environments in order to guarantee operational stability.

Example: In Figure 3 on the following page, a single manifest can deploy Kubernetes in a consistent way, providing the same functional deployment with the same releases across multiple environments. Those environments can even cross multiple IaaS providers by using the CPI abstraction. The simplified and partial deployment manifest in the image above is declaring which *BOSH stemcell*, *BOSH Release*, and config properties to use to deploy functionally identical Kubernetes clusters in two different environments.

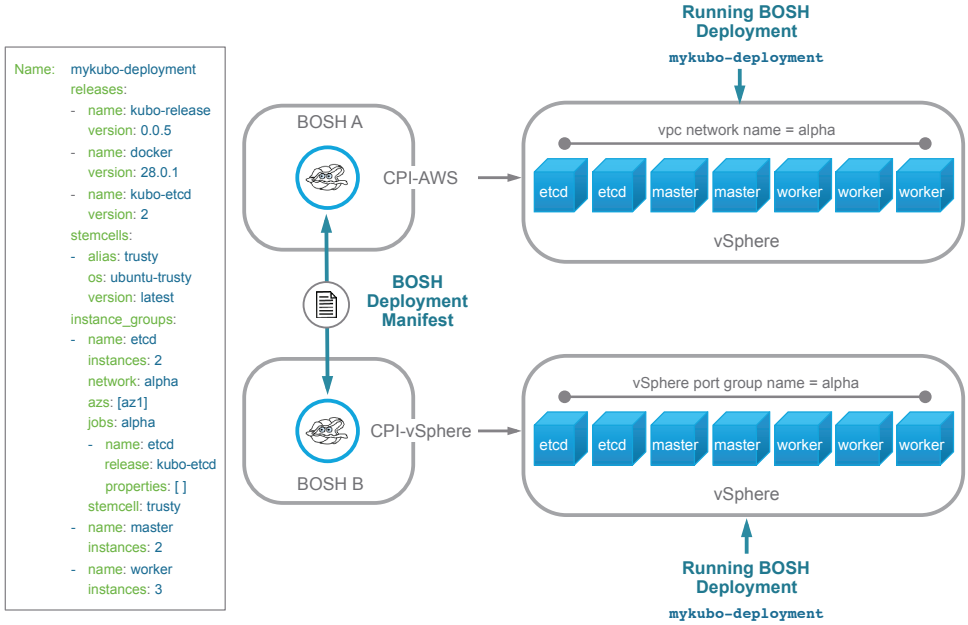

Figure 3: BOSH Reproducibility

• **Consistency**

BOSH also enforces consistency in BOSH release development to ensure that virtually any software can be packaged, versioned, and deployed in a similar pattern. This also provides operational stability.

BOSH Use Cases

BOSH's principal value lies in simplifying the deployment and day 2 lifecycle management of complex systems. It was primarily developed to deploy Cloud Foundry but has been extended by developers to deploy many other environments, both simple and complex. Systems to which BOSH can deploy can be found in two primary locations. The first is Pivotal Network, where Pivotal curates commercial BOSH releases of Pivotal Cloud Foundry as well as Pivotal services that are typically driven by Pivotal Operations Manager + BOSH. The second location is BOSH.io, which hosts an OSS community repo of various systems that can be deployed.

A prime example of a BOSH use case is Kubernetes powered by BOSH, aka Kubo.

Referencing Figure 4 on the following page, we can see the key benefits that BOSH provides the operator.

1. **Repeatability:** In a cloud native development environment, the operator can generate two or more similar deployment manifests to deploy two or more unique but functionally identical Kubernetes deployments to meet the needs of multiple developer consumers.

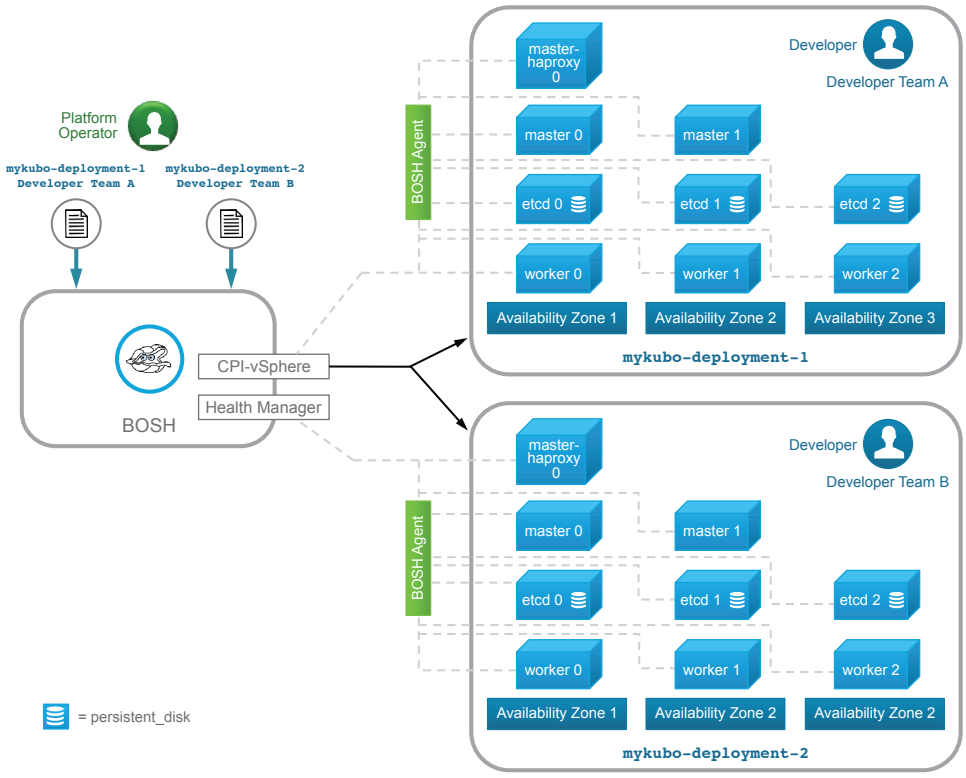

Figure 4: BOSH Use Case: Kubo

2. **Day 2 operations:** BOSH lifecycle management makes it easy to keep all of the Kubernetes deployments healthy.
   - **Maintain health:** Each VM or instance deployed by BOSH also deploys an agent that communicates health back to BOSH. If a Kubo node is unhealthy, BOSH will automatically try to repair and or rebuild the affected node. This improves uptime.
   - **Increase uptime:** Each release job instance type can have multiple VMs or instances distributed across availability zones to ensure services provided are not affected by physical faults in a given availability zone. Availability zones are only supported on certain CPIs, such as the vSphere CPI where availability zones map to vCenter clusters.
   - **Patching:** Because BOSH uses versioned releases, it is trivial for an operator to upgrade the Kubernetes Kubo release and apply it to all running deployments with little to no interruption of service. BOSH will update each deployment as well as maintain its state by: (1) detaching persistent disks, (2) rebuilding the affected VMs or instances, and then (3) re-attaching persistent disks.

Deploying BOSH

BOSH Architecture

BOSH is typically deployed as a single VM or instance. That VM/instance has many components that perform vital roles in enabling BOSH to manage deployments at scale:

- **NATS:** Provides a message bus via which the various services of BOSH can interact.
- **POSTGRESQL:** BOSH writes all of its state into a database. Typically that database is internal to a single VM BOSH deployment and provided by Postgres. This can be modified, however, to use an external data source so that the BOSH VM can be rebuilt and reconnect to the database to reload its persistent state.
- **BLOBSTORE:** Each stemcell and release uploaded to BOSH is stored in a blobstore. Default deployments of BOSH use an internal store (webdav), but, like the Postgresql database, this can also be externalized.
- **Director:** The main API that the BOSH CLI will interface with to allow an operator to create and manage BOSH deployments.
- **Health Monitor:** BOSH requires that each VM it deploys have an agent that it can communicate with to assign and deploy jobs from BOSH releases that are defined in a deployment manifest. It will also maintain the health of each VM or instance it has deployed. The agent will report vitals back to BOSH and in cases where services in the VM are faulted, or the agent is unreachable, the Health Monitor can use plugins to restart services and even rebuild the VM or instance.
- **CPI:** The CPI is the IaaS-specific executable binary that BOSH uses to interact with the defined IaaS in its deployment YAML.
- **UAA:** Provides user access and authentication that allows BOSH to authenticate operators via SAML or LDAP backends.
- **CREDHUB:** Manages credentials like passwords, certificates, certificate authorities, SSH keys, RSA keys, and arbitrary values (strings and JSON blobs). BOSH will leverage credhub to create and store key credentials for deployments, like public certificates and keys.

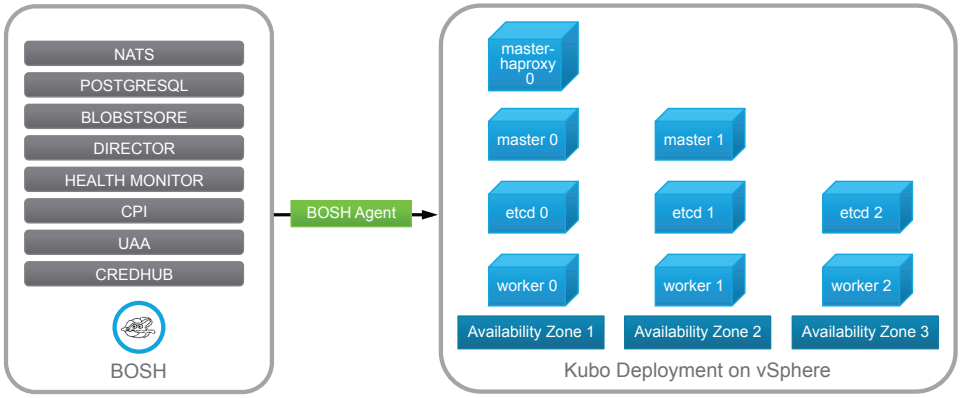

Figure 5: BOSH Architecture

BOSH References

Full reference on BOSH can be found here: BOSH.io

CookBook: How to deploy "Kubo" on vSphere

BOSH is deployed by using the BOSH CLI, passing the correct cmd line arguments, or storing those arguments as variable data within additional YAML files to define how BOSH itself will be deployed. This 'CookBook' section outlines the steps required to deploy BOSH, and offers guidance for a basic Kubo deployment.

Prerequisites include:

- vSphere vCenter 6.x
- 1 x vSphere Datastore with adequate space for deployment
- 1 x vCenter Resource pool for your Kubo deployment

Steps to deploy BOSH (for Mac OSX):

Operating system specific installation instructions for CLI are documented here.

Get the BOSH CLI

- This cookbook uses version 2 of the BOSH CLI. It is a Go compiled binary, but does have some OS dependencies.

```
1. sudo wget -O /usr/local/bin/bosh
https://s3.amazonaws.com/bosh-cli-artifacts/bosh-cli-2.0.28-darwin-amd64 && sudo
chmod 755 /usr/local/bin/bosh
```

Prepare MAC OS requirements, like Ruby

- Other OS Specific Instructions can be found here and here.

```
2. gem update --system
3. xcode-select --install
4. brew install openssl
```

Use GIT client to clone the BOSH deployment repo

- This cookbook assumes you have the git client already installed. This repo contains everything you will need to deploy a BOSH instance.

```
5. git clone https://github.com/cloudfoundry/bosh-deployment
6. cd bosh-deployment
```

Deploy BOSH

- The single CLI command below will deploy the BOSH instance. -o flags are one or more YAML files that are composed to form a single manifest. -v flags are the variables that we assign to variable markers in each YAML. The BOSH int, or interpolate, command will compose the manifest from the YAML stubs and the variables.

```
7. /usr/local/bin/bosh create-env bosh.yml \
     --state=mystate.json \
     --vars-store=mycreds.yml \
     -o vsphere/cpi.yml \
     -o uaa.yml \
     -o misc/powerdns.yml \
     -o credhub.yml \
     -v director_name=kubobosh \
     -v internal_cidr=[[CIDR-OF-NETWORK-FOR-BOSH-VM]] \
     -v internal_gw=[[GATEWAY-OF-NETWORK-FOR-BOSH-VM]] \
```

```
      -v internal_ip=[[IP-OF-NETWORK-FOR-BOSH-VM]] \
      -v network_name=[[VCENTER-PG-NAME-NETWORK-FOR-BOSH-VM]] \
      -v vcenter_dc=[[VCENTER-DC-NAME]] \
      -v vcenter_ds=[[VCENTER-DATASTORE-NAME]] \
      -v vcenter_ip=[[VCENTER-IP]] \
      -v vcenter_user=[[VCENTER-USER]] \
      -v vcenter_password=[[VCENTER-PASSWD]] \
      -v vcenter_templates=kubobosh-templates \
      -v vcenter_vms=kubobosh-vms \
      -v vcenter_disks=kubobosh-disks \
      -v vcenter_cluster=[[VCENTER-CLUSTER]] \
      -v dns_recursor_ip=[[YOUR-NETWORK-DNS]]

8. /usr/local/bin/bosh alias-env kubobosh -e [[IP-OF-NETWORK-FOR-BOSH-VM]] --ca-
cert <(/usr/local/bin/bosh int ./mycreds.yml --path /director_ssl/ca)
9. export BOSH_CLIENT=admin
10. export BOSH_CLIENT_SECRET=$(/usr/local/bin/bosh int ./mycreds.yml --path
/admin_password)
11. /usr/local/bin/bosh -e kubobosh env
```

Steps to deploy Kubo on BOSH

Use GIT client to clone the BOSH deployment repo

- This cookbook assumes you have the git client already installed. This repo contains everything you will need to deploy a Kubo with your BOSH instance.

```
1. git clone https://github.com/cloudfoundry-incubator/kubo-deployment
2. cd kubo-deployment
```

Generate BOSH cloud-config manifest and update it

- A cloud-config manifest is specific to each CPI and will map BOSH constructs (like availability zones) to vSphere ones. A deployment manifest will reference constructs from the cloud-config.

```
3. /usr/local/bin/bosh int configurations/vsphere/cloud-config.yml \
      -o manifests/ops-files/k8s_master_static_ip_vsphere.yml \
      -v director_name=bosh \
      -v internal_cidr=[[CIDR-OF-NETWORK-FOR-KUBO-CAN-BE-SAME-AS-BOSH]] \
      -v internal_gw=[[GATEWAY-OF-NETWORK-FOR-KUBO-CAN-BE-SAME-AS-BOSH]] \
      -v internal_ip=[[DNS-OF-NETWORK-FOR-KUBO-CAN-BE-SAME-AS-BOSH]] \
      -v kubernetes_master_host=[[IP-FOR-KUBO-MASTER-VIP]] \
      -v reserved_ips=[[IP-RANGE-YOU-DONT-WANT-BOSH-TO-USE ex: 192.168.100.10-
192.168.100.20]] \
      -v network_name=[[VCENTER-PG-NAME-NETWORK-FOR-KUBO-CAN-BE-SAME-AS-BOSH]]
\
      -v deployments_network=[[VCENTER-PG-NAME-NETWORK-FOR-KUBO-CAN-BE-SAME-AS-
BOSH]] \
      -v vcenter_cluster=[[VCENTER-CLUSTER-FOR-KUBO]] \
      -v vcenter_rp="[[VCENTER-RESOURCE-POOL-FOR-KUBO]]"
       > mycloudconfig.yml
4. bosh -e kubobosh update-cloud-config mycloudconfig.yml
```

Generate Kubo deployment manifest

```
5. /usr/local/bin/bosh int manifests/kubo.yml \
     -o manifests/ops-files/master-haproxy-vsphere.yml \
     -o manifests/ops-files/worker-haproxy.yml \
     -v deployments _ network=[[VCENTER-PG-NAME-NETWORK-FOR-KUBO-CAN-BE-SAME-AS-
BOSH]] \
     -v kubo-admin-password="mykubopasswd" \
     -v kubelet-password="mykubopasswd" \
     -v kubernetes _ master _ port=443 \
     -v kubernetes _ master _ host=[[IP-FOR-KUBO-MASTER-VIP]] \
     -v deployment _ name=mykubocluster \
     -v worker _ haproxy _ tcp _ frontend _ port=1234 \
     -v worker _ haproxy _ tcp _ backend _ port=4231 > mykubo.yml
```

Upload latest BOSH stemcell

```
6. /usr/local/bin/bosh -e kubobosh upload-stemcell https://s3.amazonaws.com/
bosh-core-stemcells/vsphere/bosh-stemcell-3421.11-vsphere-esxi-ubuntu-trusty-go _
agent.tgz
```

Upload BOSH Kubo release

```
7. wget https://github.com/cloudfoundry-incubator/kubo-release/releases/download/
v0.0.5/kubo-release-0.0.5.tgz
8. /usr/local/bin/bosh -e kubobosh upload-release kubo-release-0.0.5.tgz
```

Deploy Kubo :)

```
9. /usr/local/bin/bosh -e kubobosh -d mykubocluster deploy ~/kubo-deployment/
mykubo.yml
```

The latest version of this document can be found here.

About VMware
VMware delivers infrastructure technologies enabling developers and IT to build, run and manage cloud-native applications to increase business agility, security and efficiency.

Visit www.vmware.com/solutions/cloudnative.html for more information.

Twitter link: https://twitter.com/cloudnativeapps

Blog link: https://blogs.vmware.com/cloudnative/

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com