Last Updated: March 4, 2020

ADM Nimbus and Blueshift

2020 Partner Boot Camp - Hands-on Guide

Based on NimbusServer 2019-R1 and NimbusClient 2020-R1

Contents

Welcome!

These exercises will guide you through the essentials of the Nimbus demo environment. This guide and the associated exercises are for individuals new to Nimbus or who are looking for a quick refresher.

Nimbus is an easy-to-use Docker-based demo environment built around the ADM product pillar. It is the brainchild of Bryan Cole, who architected the framework, and was meticulously constructed by Bryan Cole, Mark Steffensen, Jason Hrabi and Dave Flynn, with help from Andreas Widmann.

Prior to Nimbus a number of other demo environments existed, both locally and in the cloud. These systems typically had issues with connectivity, cost and maintaining the latest product versions. The size of local all-in-one images had grown to over 50 GBs and keeping everything up-to-date and integrated was a daunting task.

Nimbus changes the way this works. Instead of a single virtual machine containing lots of installed software products, each product is implemented as a Docker container or a set of Docker containers. Users can pull down just the images they need and the VM itself (a Linux VM called NimbusServer) is just a home for Docker and its images and containers. Separate, smaller Windows-based VM's (NimbusClient and NimbusWindows) are used to house the ADM products that run only on Windows or that run as server-class applications on Windows only (e.g., UFT, LR, SV Designer, Silk Central, etc.)

Roles

Nimbus is designed to primarily be used by presales and ADM partners. However, teams in technical marketing, R&D, consulting, support and SaaS may find Nimbus useful as well.

- Presales
- Partners
- Technical Marketing
- R&D
- Consulting
- Support
- SaaS

Activities

You'll be following this progression of learning activities:

- Getting Started / Basic Linux and Docker Commands
- Starting the AOS Containers
- Starting the DevOps Container

Let's get started!

EXERCISE 1: Getting Started with Nimbus

Nimbus is a demo environment and NOT meant for production use. It utilizes freeware or open source software for its repositories, file systems and operating systems making them unsuitable for production use. It is, however, a very robust platform for demos, product configuration and product debugging.

Nimbus is also available as downloadable VMs running on a host laptop or in the cloud on Amazon AWS and Microsoft Azure (limited availability).

Nimbus consists of three virtual machines, **NimbusServer**, **NimbusClient** and **NimbusWindows**, along with a number of Docker images. For laptop hardware prerequisites for using these VMs see the Appendix.

1. **NimbusServer** is a CentOS Linux VM running Docker. It is used to pull Docker Images, such as ALM Octane, ALM .NET and DevOps, from the public Docker Hub repository and run them as Docker containers. The following Docker Images (and more) are currently available:

Dockerized Application Images		
ALM Octane	Advantage Online Shopping (AOS)	SonarQube
ALM .NET	DevOps	Dimensions CM
UFT Mobile	IntelliJ	Service Virtualization Manager
Agile Manager	Software Security Center (SSC)	Network Virtualization
AutoPass	Source Code Analyzer (SCA)	PPM

2. **NimbusClient** is a Microsoft Windows (Windows Server 2016) VM containing products such as Unified Functional Testing, Service Virtualization and LoadRunner, which run only on a Windows Platform. This image is periodically updated as new products versions come out.

NimbusClient Windows Applications		
UFT One	SV Designer	Microsoft Word 2013
LR Controller	Sprinter	Microsoft Excel 2013
LR VuGen	NV for LR	Reflections TE
LR Analysis	Internet Explorer (for ALM)	

3. **NimbusWindows** is a Microsoft Windows (Windows Server 2016) VM containing server-based products such as Silk Central, RPA Recorder and Robot, Dimensions RM and Release Control, which run only on a Windows Platform. This image is periodically updated as new products versions come out.

NimbusWindows Windows Applications		
Solution Business Manager	Silk Central	Network Virtualization
RPA Recorder	Silk Meter	Serena License Manager
RPA Robot	Dimensions RM	
Release Control	UFT Mobile Connector	

The combination of these virtual machines plus a library of Docker images creates a compelling demo platform that we call Nimbus. This platform offers an extremely flexible and efficient stage for ADM demos.
As these products evolve, each product release is built and tested as a new Docker image. Acquiring the latest Docker release of an application is a simple matter of pulling it from the public Docker Hub and running it.

Partners should note that by using the Nimbus environment they agree to all contained or embedded licenses and any other open source or freeware licenses and agreements. Partners who download the NimbusClient or NimbusWindows VM's are also responsible for applying a valid Windows Server 2016 Standard license to that image since it is provided without a valid Microsoft license.

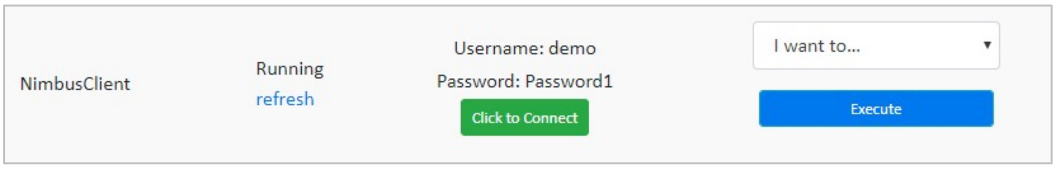
Connecting to Blueshift

If you don't have the Nimbus environment locally, the technical enablement team will provision a remote environment (Blueshift) for you. Blueshift is a remote demo environment hosted on Amazon's AWS. Blueshift utilizes a web browser which connects to an Apache Guacamole server utilizing port 443.

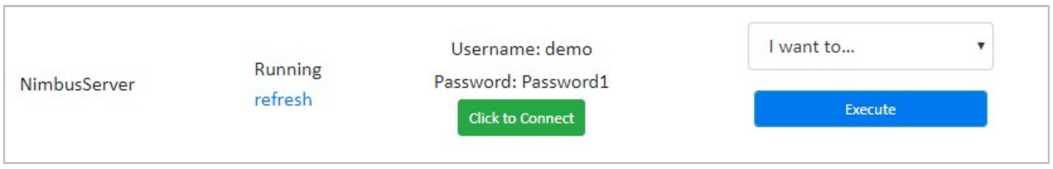
The following instructions relate to using Blueshift as your demo environment.

1. You will receive your login username and password from your instructor.
   Connect to this website and **login** with your provided credentials.
   https://www.mfdemoportal.com/ **adm_enablement** **Brazil2020**
2. From the Blueshift main page, select **My Orders** at the top.
3. Locate your order (this was created for you) and click the link.
4. From the **Order Information** page, you'll see three **Click to Connect** links.
   Click each of these to open a new browser tab with a graphical desktop displayed.

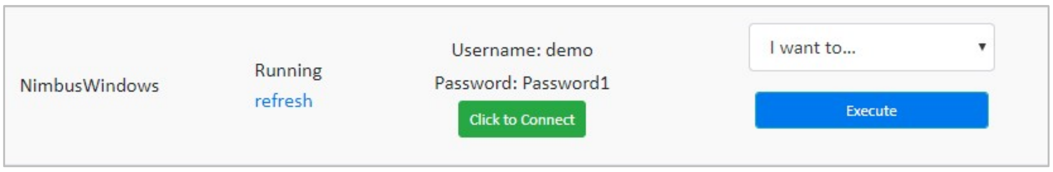
   a. NimbusClient – This **Windows** desktop image contains tools that run only on Windows.


NimbusClient
Running
Username: demo
Password: Password1

   b. NimbusServer – This **Linux** desktop contains Docker and some downloaded docker images.


NimbusServer
Running
Username: demo
Password: Password1

   c. NimbusWindows – This **Windows** desktop contains server applications on Windows.


NimbusWindows
Running
Username: demo
Password: Password1

5. Once you are connected you can treat the browser window just like a normal desktop**.**

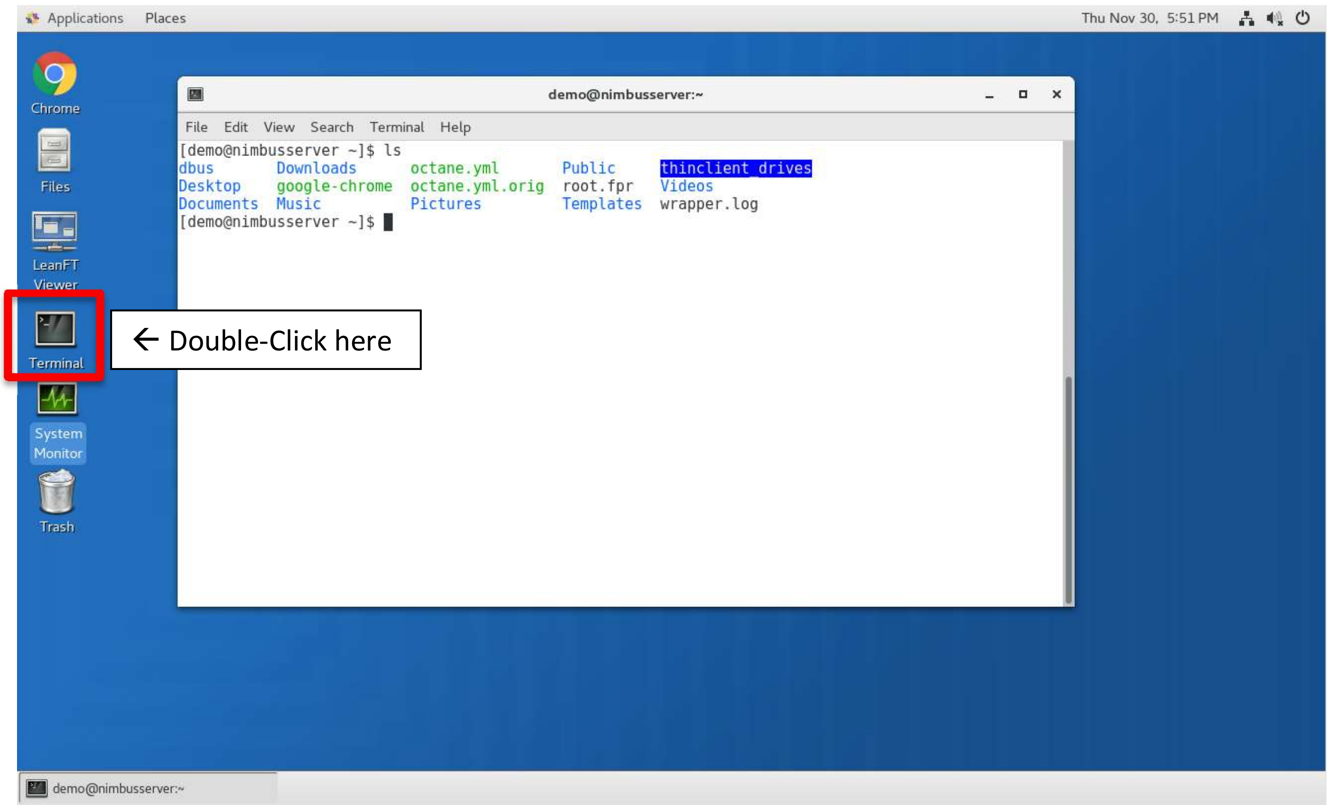
Disconnecting

You should never shutdown an image but instead, always disconnect by **closing the browser tab**. If you shutdown or power off your image you may lose any changes you made to the image.

Running Commands from a Terminal

Your NimbusServer machine consists of a Centos Linux operating system along with a graphical UI called Gnome. This UI is your desktop interface and contains icons to start your applications. Linux is often used without a graphical display, interacting just with a set of commands in a **Terminal** window.

Open up a **Terminal** by clicking the Terminal icon on your desktop. This is similar to the Command window used on Windows and allows you to type commands into Linux.



```
[demo@nimbusserver ~]$ ls
dbus       Downloads      octane.yml       Public      thinclient_drives
Desktop    google-chrome  octane.yml.orig  root.fpr    Videos
Documents  Music          Pictures         Templates   wrapper.log
[demo@nimbusserver ~]$ 
```

For some fun, type the following commands:

```
$ sudo yum install sl

$ sl
```

What do you think the **sl** command is short for?

____

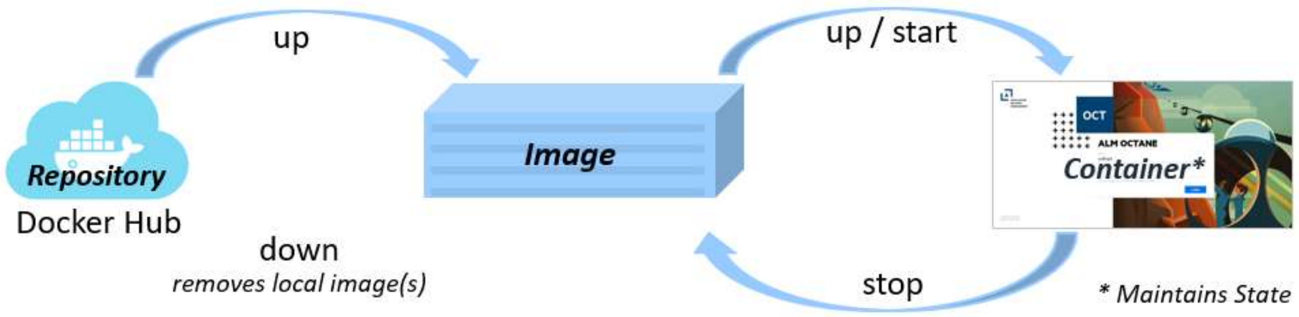
Docker Images versus Virtual Machines

If you've used virtual machines before it's important to understand the differences between Docker images and virtual machines. Both can save their state and both can revert to an initial state. However, Docker images share a common core OS, thus optimizing performance and overall disk usage.

In a typical VM environment (like we use for NimbusClient), we install a number of applications onto an operating system. It isn't possible to run two versions of the same software at a time because there's just one operating system and one file system.

With containerized applications, the software is installed on a common Linux platform. When you run multiple images (called containers) they each share the same operating system but essentially have their own file system. Docker is able to split these systems into layers and share some of them, thus optimizing use. For our Nimbus environment we chose to use CentOS (the free version of RedHat) which also reduces the overhead.

The following diagram show how Docker images are pulled from a repository and become resident locally. Then, when they "start" they create a "container" which can be stopped and started at will while maintaining state.



Nimbusapp versus Docker's Pull or Run

Docker provides two commands to get an image from a repository – pull and run. When you execute a "docker pull …" command, Docker first looks at the local Docker repository (on your local disk) and sees if the requested image exists there. If it does, then nothing is done – it's already been pulled.

Instead of using the docker pull and run commands (which can be quite lengthy to type) we've created a script called "nimbusapp" which simplifies these operations and allows us to define dependencies and run multiple containers with one, simple command.

When you execute a "nimbusapp … up" command it combines the "pull" function along with the "start" function and runs the image after all files are locally downloaded. An image that has been "run" creates a "container" which is a live, running process, including a file system.

After an image is on your local disk and has been "run" you can just use a "nimbusapp … start" command to start a stopped container again. You can use "docker images" along with "docker ps" or "docker ps –a" to see which images and containers are available. You can stop a container with "nimbusapp … stop " or restart it with "nimbusapp … restart".

Basic Linux Commands

Below is a chart of some basic Linux commands for your reference.

Command	Action	Command	Action
cd	Change directory	**tail**	Show last 10 lines of a file
ls	List files	**find**	Locate a file on the system
pwd	Present working directory	**ps**	Process status
whoami	Who am I logged is as?	**rm**	Remove a file (or directory)
su	Switch user	**exit**	Logout/exit a terminal session
cat	Short for concatenate	**echo**	Send text to the terminal
ping	Package Internet Groper	**ifconfig**	Display networks
mkdir	Make a directory	**chmod**	Change file/directory mode
touch	Update a file timestamp	**chgrp**	Change file/directory group
curl	Transfer data/files	**yum**	Package installer

If you're very comfortable with Linux you can skip this section and move on to Basic Docker Commands.

Run each of the following commands to see what they produce (just type what comes after the **$** prompt):

1. `$ ls`
   This displays a list of files in your current directory.

2. `$ ls -al`
   This displays a more detailed list of files including which are directories (d) and which have read/write/execute (r-w-x) permissions for the owner, group and all other accounts.

3. `$ pwd`
   This displays your present working directory. It should display this path, for example: **`/home/demo`**
   This "home" location (for the demo user) is also referred to as ~ (tilde).

4. `$ cd ~/Desktop; ls`
   This command actually runs two commands (; separates them) and should show a few files and your prompt will change to show that you're now in the Desktop directory. **Note: Linux is case-sensitive!**

5. `$ sudo find /etc -name hosts -print`
   This command logs you in searches from the slash (/) or root directory and then "finds" all files (starting at specified directory) that are named "hosts" and prints the result. You can also use wildcards in the search. This is a useful command to find the path to files you know are on the disk.

Simplified Docker Commands

Linux folks love to shorten and simplify the commands they run regularly. We've helped with this "simplification" by creating some aliases and functions which are stored in your home directory's .bashrc file. This file is run whenever you open a terminal and establishes some shortened commands.

For example, instead of typing "docker images" to see what images you have locally you can just type "di" instead. Type the following command from your NimbusServer terminal window:

```
$ di
```

You should see a long list of images that have already been pulled to your local machine.
Here's another couple of command you should try:

```
$ docker ps
$ dps
```

The first command shows a list of running container, in this case autopass, but the data wraps around the screen and it hard to view unless you stretch out your window. The second version shows much of the same information but in a more compact view.
Now, try this command:

```
$ de autopass
```

This command is equivalent to "docker exec –it autopass /bin/bash". It allows you to enter the file system for the autopass container and then you can move around and see what files exist there. If you wanted to go inside other containers just substitute that container name in place of "autopass" like "de octane".
Next, type the following commands:

```
$ pwd
$ ls -al
```

As mentioned above, pwd shows you what directory you are in and the second command lists the files in that directory including hidden files.
Finally, type this command to exit your journey into the autopass container:

```
$ exit
```

To further simplify our Nimbus environment we've created the "**nimbusapp**" command. This command uses docker-compose and docker-app, two other docker commands to define container dependencies and the order in which containers start. Nimbusapp greatly simplifies the startup of multiple containers.
Type the following and notice how many containers are started:

```
$ nimbusapp sv start
```

For now, run the following command to stop these containers:

```
$ nimbusapp sv stop
```

Logging in to Docker

1. Whenever you want to interact with Docker hub (pull, run, push, etc.) you need to be logged into the Docker Hub Repository. You'll need Internet access and your Docker Hub ID and password to do this.

   From a Linux terminal window, login in to Docker Hub using your **Docker ID** and password.
   If you don't have a Docker ID, go to https://hub.docker.com and create a free one.

   ```
   $ docker login
   $ Username: <your docker ID>
   $ Password: <your docker hub password>
   ```

   If everything is correct, you will receive the `Login Succeeded` message.

2. Check your local images again by typing: `$ di`

3. Let's also check your local containers by typing: `$ dpsa`
   This command will show all current containers both running and stopped (i.e. docker ps –a).

4. Open a Chrome browser and log in to https://hub.docker.com and examine your available images.

Bonus Question #1

How many docker images are in your Docker environment?

____

Bonus Question #2

How many docker containers are currently running in your Docker environment?

____

Challenge Exercise #1

Try running several of the Basic Linux Commands and then determine what the following command does:

```
$ cat /etc/hosts | grep nimbusserver
```

____

Awesome!

You have just learned how to use Nimbus!

Exercise 2: Starting the AOS Containers

The Advantage Online Shopping site (AOS) is a mock e-commerce site to be used for testing. There is a production site located at http://advantageonlineshopping.com as well as a locally available AOS that you can run as a set of Docker containers. In this exercise we'll start up a copy of the AOS website and verify that it's working properly.

Starting the AOS Containers

1. First we need to start the **aos_postgres** container which holds the Postgres database for AOS. From a NimbusServer terminal window type the following to start the **aos_postgres** container:

   ```
   $ nimbusapp aos start
   ```

2. Verify that your container is starting properly by typing:

   ```
   $ dps
   ```

3. Advantage Online Shopping actually consists of the following 7 separate code modules:
   - accountservice
   - catalog
   - MasterCredit
   - order
   - ROOT
   - SafePay
   - ShipEx

   The entire codebase for AOS is available and we can edit or change how it works. When we edit the code (like changing the Speakers category to Audio) we're actually editing code in the ROOT module which gets redeployed via a job in Jenkins (AOS_Web_Deploy_Root).

   To create the AOS web application we can just copy these directories/files to a directory inside of a Tomcat web server and then start Tomcat and it does the rest.

   AOS uses three containers to run:

• aos-main	Contains a Tomcat web server and most of the modules
• aos-postgres	Contains the AOS database
• aos-accountservice	Contains just the accountservice module

   The aos-accountservice module was deployed separately so that we can virtualize this service with Micro Focus Service Virtualization.

4. Wait for AOS to be fully booted (a couple of minutes) and check its status using the **System Monitor** on your desktop. Initially you'll see a lot of CPU activity that will eventually calm down. At that point your application has stabilized. Verify that AOS has started properly by opening Chrome at this URL:

   ```
   http://nimbusserver.aos.com:8000/#/
   ```

   You can also use the Chrome **AOS** browser shortcut to get there.

> **NOTE:** Typically, running the AOS application would require having all the files located on one Tomcat instance but we've split this into two, **aos_accountservice** and **aos_main**. You can actually run AOS with just the **aos_postgres** container and **aos_main** and it will appear to work fine. However, if you run it without starting **aos_accountservice** you won't be able to create new accounts or login with an existing account.

Bonus Question #3

Which directory on the aos_main container holds all the different modules?
HINT: Use `$ de aos-main` to get inside the container and then

```
$ cd /; find . -name ShipEx -print
```

____

Challenge Exercise #2

Shutdown the aos-accountservice container and find out what functionality no longer works.
Use this command to stop the aos_accountservice:

```
$ docker stop aos-accountservice
```

What happens if you try and set the Country when creating a new account?

____

NOTE: Don't forget to run "docker start aos_accountservice" when you're done with this experiment.

Fantastic!

You have just started Advantage Online Shopping locally!

Exercise 3: Starting the DevOps Container

The DevOps container is a very special container since it houses the source code for AOS as well as the Jenkins server, the CI tool that builds and tests AOS. It forms the crux of our DevOps story and is key to any DevOps discussion, presentation or demo.

Starting the DevOps Container

1. First we need to start the **DevOps** container which holds the AOS source code and the Jenkins server. From a NimbusServer terminal window type the following to start the **DevOps** container:

   ```
   $ nimbusapp devops start
   ```

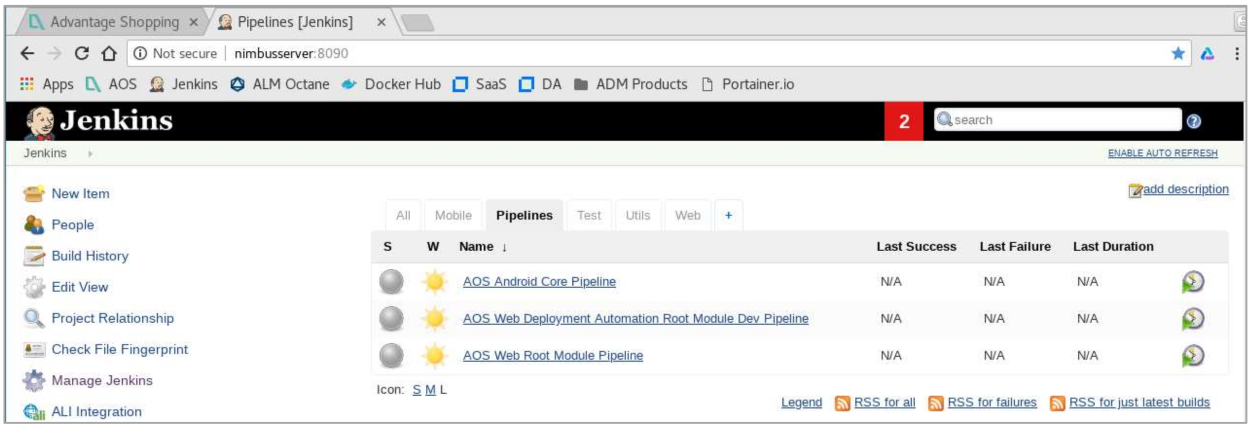
2. Verify that your container started properly by typing:

   ```
   $ dps
   ```

3. Open a Chrome browser and navigate to the following URL (or use the **Jenkins** browser shortcut). You may need to wait a minute or two for Jenkins to finish starting up.

   ```
   http://nimbusserver:8090
   ```

4. You should see the **Jenkins** home page as shown below:



S	W	Name ↓	Last Success	Last Failure	Last Duration
		AOS Android Core Pipeline	N/A	N/A	N/A
		AOS Web Deployment Automation Root Module Dev Pipeline	N/A	N/A	N/A
		AOS Web Root Module Pipeline	N/A	N/A	N/A

A bit about Jenkins…

Jenkins is a commonly used CI (Continuous Integration) system. It consists of a web server that runs the Jenkins user interface and a collection of "build jobs" that were originally just compilation tasks. Jenkins has since evolved into an advanced execution engine that has thousands of specialized plug-ins.

There are a number of different CI tools but Jenkins is by far the most recognized and used (it's free, after all). Some of these other tools are Bamboo, TeamCity and Go (formerly Cruise Control).

At its core, Jenkins executes a defined set of commands, typically from a command line. In the image above, there are a number of tabs:

- **All** – this tab contains every job in Jenkins, whether a pipeline or a single build.
- **Mobile** – this tab contains just jobs related to mobile and building the AOS Android app.
- **Pipelines** – this tab contains just pipeline builds (combinations of other builds).
- **Test** – this tab contains just builds related to testing (regression and Gherkin).
- **Utils** – this tab contains just a single build that updates the Octane API keys in Jenkins.
- **Web** – this tab contains just AOS web build jobs.

Click on each of these tabs to see the kinds of jobs that each contains.

A build job in Jenkins is typically a series of commands that copy files, compile source code, deploy build artifacts and spin up Docker containers. There's plenty more you can do as well.

For our AOS DevOps story the AOS Web pipeline executes the following steps:

I. **Build** the ROOT module of the AOS Web application using Maven.
II. **Undeploy** the current ROOT module file from the aos_main container.
III. **Deploy** the new ROOT module into the aos_main container.
IV. **Run** a basic UFT Developer regression test to validate the new AOS application.

During one of these steps, code is checked out from a local Git repository (also on the DevOps container) and then built using Maven (a popular code build and configuration tool). These steps are easily changed or extended but they show how we integrate with a full DevOps process. Plus, you can run all of this on a laptop in just a couple of minutes!

Wonderful!

You have just started the DevOps container with Jenkins!

Appendix

Computer Prerequisites

The NimbusServer, NimbusClient and NimbusWindowsVM images do have some performance and memory requirements. If your hardware is limited (e.g. 8-16 GB of RAM), we suggest you only run the containers you need to reduce memory consumption. We also suggest stopping other programs on your laptop, such as Outlook, Skype and anything else unnecessary. **Your laptop should have at least 16 GB's of memory installed.**

NimbusServer – 2.2 GB download			NimbusClient or NimbusWindows – 24 GB download		
Memory	12GB+ free	16GB+ preferred	**Memory**	8GB+ free	8GB preferred
CPU	Intel i5 >1.70 GHz	Intel i7 preferred >2.40 GHz	**CPU**	Intel i5 >1.70 GHz	i7 preferred >2.40 GHz
Disk Space	50-100GB free	SSD preferred	**Disk Space**	35-80GB free	SSD preferred

Partner Virtual Machine Download

The NimbusServer and NimbusClient VMs are available for download as compressed images from the Technical Enablement FTP server at ftp.ext.hpe.com. The images are multi-part archives that use the free 7-Zip utility located in the Nimbus directory for your use. The FTP login credentials are shown below:

- FTP site: ftp-pro.houston.softwaregrp.com
- Account: prtnr_ro
- Password: dn+JT4rn
- Directory: /VMWare Images/NimbusDemoEnvironment

Special Note

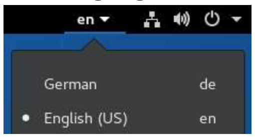
When starting either VM for the first time, make sure you have **virtual technology enabled** in your BIOS (VTx) and remember to select **I moved it,** and not the default **I copied it,** when prompted by VMware Workstation or Player. Failure to do this may cause license issues.

Changing your keyboard language layout

NimbusServer image:

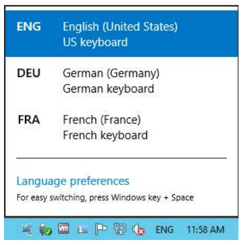
1. From the **NimbusServer** desktop select the **Applications>System Tools>Settings**.

2. Select **Region & Language**, then under **Input Sources** click the + sign and select your preferred language.
3. Click the keyboard icon to verify the new keyboard layout.
4. You'll notice a new icon in your system tray on the top right. It will show the abbreviation of the country and/or language name that is currently selected. Click the icon to change the language.



NimbusClient or NimbusWindows Image:

1. Open the control panel and navigate to: **Control Panel>Clock, Language, and Region>Language**
2. Click **Add Language** and select your preferred language.
3. Close the windows and you'll notice a new icon in your system tray on the bottom right.
4. Left-Click the language to select your preferred language.


ENG English (United States) US keyboard
DEU German (Germany) German keyboard
FRA French (France) French keyboard
Language preferences
For easy switching, press Windows key + Space
ENG 11:58 AM

Notes

Micro Focus International plc
The Lawn, 22-30 Old Bath Road, Berkshire, RG14 1Q