



Learning VMware Workstation Pro for Windows: Volume 2

Implementing and Managing
VMware's Desktop Hypervisor
Solution

—
Peter von Oven



Apress®

Learning VMware Workstation Pro for Windows: Volume 2

**Implementing and Managing
VMware's Desktop Hypervisor
Solution**

Peter von Oven

Apress®

Learning VMware Workstation Pro for Windows: Volume 2: Implementing and Managing VMware's Desktop Hypervisor Solution

Peter von Oven
FAIRFORD, UK

ISBN-13 (pbk): 979-8-8688-0863-0

ISBN-13 (electronic): 979-8-8688-0864-7

<https://doi.org/10.1007/979-8-8688-0864-7>

Copyright © 2024 by Peter von Oven

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Aditee Mirashi
Development Editor: James Markham
Coordinating Editor: Kripa Joseph

Cover designed by eStudioCalamar

Cover image by Unsplash (www.unsplash.com)

Distributed to the book trade worldwide by Apress Media, LLC, 1 New York Plaza, New York, NY 10004, U.S.A. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at <http://www.apress.com/bulk-sales>.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub (<https://github.com/Apress>). For more detailed information, please visit <https://www.apress.com/gp/services/source-code>.

If disposing of this product, please recycle the paper

*As always, this book is dedicated to my family
who have again supported me along the way while
I have been busy writing.*

Table of Contents

- About the Authorxi**
- Introductionxiii**

- Chapter 1: Working with Containers..... 1**
 - What Is a Container? 1
 - Getting Started with the vctl Command 3
 - What Is the vctl Command..... 3
 - vctl Command Prerequisites..... 5
 - Setting Up and Managing a Container Runtime 6
 - Updating the CRX VM and Kubernetes Node Configuration..... 11
 - vctl Commands..... 14
 - The build Command..... 14
 - The completion Command..... 16
 - The create Command 17
 - The describe Command..... 20
 - The exec Command 21
 - The execvm Command 23
 - The help Command..... 24
 - The images Command..... 26
 - The inspect Command..... 28
 - The kind Command..... 29
 - The login Command..... 29
 - The logout Command 30

TABLE OF CONTENTS

The ps Command.....	31
The pull Command.....	32
The push Command.....	34
The rm Command.....	36
The rmi Command.....	37
The run Command.....	38
The start Command.....	43
The stop Command.....	45
The system Command.....	46
The tag Command.....	47
The version Command.....	48
The volume Command.....	48
Summary.....	49
Chapter 2: Using the vmware Command.....	51
vmware Command Syntax.....	52
Command Options.....	53
Show Program Version.....	53
Powering on a Virtual Machine.....	54
Powering on a Virtual Machine in Full Screen.....	56
Start Virtual Machine in Paused Mode.....	57
Close a Virtual Machine on Power Off.....	59
Set a Virtual Machine Variable.....	61
Open a New Window.....	63
Launch in Full Screen Mode.....	67
Console Connections.....	71
Display the Command Options.....	74
Shortcut Example.....	76
Summary.....	82

Chapter 3: Using the vmrun Command.....	85
How to Use the vmrun Command.....	86
Running Authentication Flag Commands	88
Guest OS Commands.....	89
Power Commands	117
Snapshot Commands	122
Host Network Commands.....	126
General Commands	131
Summary.....	138
Chapter 4: VMware Workstation Pro REST API.....	141
What Is the REST API?.....	141
REST API Standard Commands.....	142
Workstation Pro API Categories	142
Setting Up the REST API	143
Connecting to the API Service	145
Configuring HTTP API Access.....	146
Workstation Pro API Explorer	147
Configuring HTTPS API Access	149
Configuring API Calls	154
Host Network Management API Calls	155
Virtual Machine Management.....	166
VM Network Adapter Management.....	177
VM Power Management	188
VM Shared Folders Management	191
Summary.....	197

TABLE OF CONTENTS

Chapter 5: Support and Troubleshooting199

- Running the Support Script 199
- Gathering Debugging Information from a VM 200
- Gathering Debugging Information for the Host 212
- Run the Support Script from a Command Prompt..... 216
- Register and Create a Support Request 218
- Troubleshooting VM Performance Issues 220
- Other Support Sources 223
- Summary 225

Chapter 6: Workstation Player227

- What Is a Workstation Player?..... 227
- Workstation Player Requirements 228
- Downloading Workstation Player..... 228
- Installing Workstation Player 232
- Workstation Player User Interface 244
- Creating a New Virtual Machine 262
- Adding an Existing Virtual Machine 274
- Managing an Existing Virtual Machine 277
- Summary 279

Chapter 7: Creating Alternative OS VMs281

- Building an Ubuntu Virtual Machine 281
- Building a Proxmox Virtual Machine..... 293
 - Installing VMware Tools for Proxmox..... 308
- Summary 312

Chapter 8: Unattended Installation	313
Installing Workstation Pro Using the Command Line	314
Prerequisites	314
Uninstalling Using the Command Line.....	315
Command Line Switch Options	317
Extracting the MSI File for Installation	319
Summary	323
Chapter 9: What's New	325
Workstation Pro Releases	325
Workstation Pro 17.0.2	326
Workstation Pro 17.5	326
Workstation Pro 17.5.1	327
Workstation Pro 17.5.2	327
Broadcom Ownership.....	328
Summary	328
Index.....	331

About the Author



Peter von Oven is an experienced technical consultant working closely with customers, partners, and vendors in designing technology solutions to meet business needs and deliver outcomes. During his career, Peter has presented at key IT events such as VMworld, IP EXPO, and various VMUG and CCUG events across the UK. He has also worked in senior presales roles and presales management roles

for Fujitsu, HP, Citrix, and VMware and has been awarded VMware vExpert for the last ten years in a row, vExpert EUC for the last three consecutive years, vExpert Desktop Hypervisor, and more. He recently became part of the new Omnisia Tech Insiders program.

In 2016, Peter founded his own company specializing in application delivery. Today, he works with partners and vendors helping drive and deliver innovative technology solutions. He is also an avid author, having now written 19 books and made numerous videos about VMware end-user computing solutions. In his spare time, Peter volunteers as a STEM Ambassador, working with schools and colleges, helping the next generation develop the skills and confidence in building careers in technology. He is also a serving Royal Air Force Reservist, recently becoming a commissioned officer working with the Air Cadet organization.

Introduction

The VMware Workstation Pro solution is a type 2 desktop hypervisor solution that enables you to run virtual machines, containers, and Kubernetes clusters on your local PC or laptop. It provides the ideal solution for building and testing virtual machines locally before moving them into production.

This second book will focus on some of the more advanced features of VMware Workstation Pro.

Throughout this book, we will work proactively with the Workstation Pro solution to enable you to build and manage virtual machines and containers locally, using step-by-step instructions with real-life screenshots to demonstrate each key feature and how it works.

We start with a high-level overview of how VMware Workstation Pro works with running containers starting with a brief overview of how containers differ from virtual machines. Then, we will look at how to set up and configure VMware Workstation Pro for running containers and then take a deep dive into the **vctl** command that is used for managing container environments.

Following on from managing and working with containers, we will look at the **vmware** command for running Workstation Pro from the command line of the host machine.

In the next couple of chapters, we will look at the **vmrun** command, a command line utility available in Workstation Pro that allows you to control virtual machines and automate guest operating system actions on virtual machines, and then the Workstation Pro API.

INTRODUCTION

After we have taken a look at support and troubleshooting tips, we will take a look at VMware Workstation Player, a cut down version of Workstation that is primarily used for running preconfigured virtual machines.

In the previous *Workstation Pro* book, Volume 1, we installed Windows-based operating systems as well as a completed vSphere cluster. In this second part, we are going to install a Linux-based virtual machine as well as an alternative hypervisor (Proxmox in this example).

The penultimate chapter will build on the installation of Workstation Pro that we covered in Volume 1, but in this second part, we will look at how to perform an unattended installation.

Finally, since Volume 1 of the book was published last year (2023), we are going to discuss all the new updates and features that have been launched since that book was published.

CHAPTER 1

Working with Containers

In this first chapter, although this book focuses on VMware Workstation Pro as the desktop hypervisor for hosting virtual machines, we are also going to look at one of the newer features that was introduced in Workstation Pro 16, and that is the ability to manage containers.

The ability to run containers in Workstation Pro started with the project Nautilus back in January 2020. [Project Nautilus](#) was a tech preview feature included in VMware Fusion, the Apple Mac version of Workstation Pro. It then became available in Workstation Pro version 16 controlled and managed by using the new `vctl` command line utility.

Before we start to get into the specific details of running containers, we are first going to take a step back and describe what a container actually is and how it differs from a virtual machine.

What Is a Container?

A question that often gets asked is “So how does a container differ from a virtual machine or hypervisor-based solution?” and why are we discussing this in a book that is all about a desktop hypervisor?

In this section, we are going to briefly compare containers and virtual machines just to serve as a reminder to what we originally discussed in the first book and to set the scene for this chapter.

As we have previously discussed, a virtual machine runs a full-blown guest operating system instance that shares the physical resources of the host machine on which it runs. It is a guest on the host's hardware. In the case of this book, Workstation Pro provides the hypervisor layer on top of an existing operating system, referred to as a type 2 hypervisor.

In contrast, a container is an environment that runs an application that is not dependent on a guest operating system. Instead, it isolates just the application by bundling (or containerizing) the application's code together with the related configuration files and libraries and with the dependencies that the application requires for it to run.

The diagram in Figure 1-1 shows the comparison between a hypervisor-based VM solution and a containerized environment.

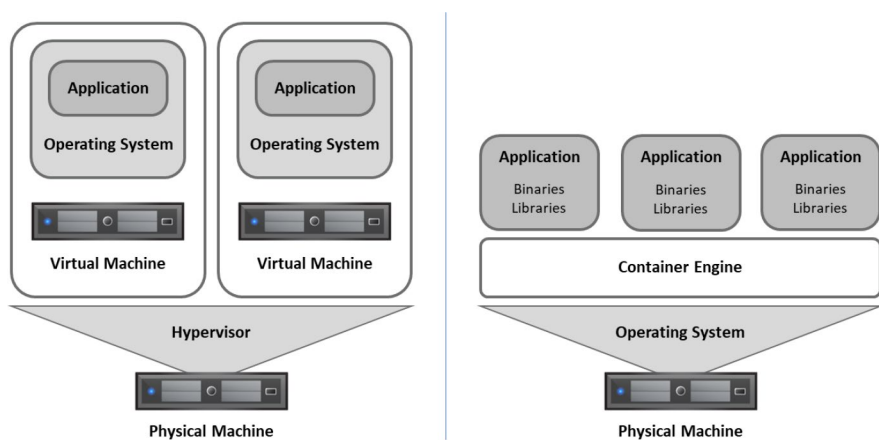


Figure 1-1. Comparison Between a Hypervisor and a Container

This bundling is where the container name originates from. If you think about moving house and using a shipping container into which you place all your belongings, by doing this, it makes it easy to move everything

around in one hit. It is the same for applications. In this analogy, substitute your household belongings for the application runtimes, config files, and libraries and there you have your containerized apps, ready to run across environments.

Containers provide a straightforward way to build, test, deploy, and redeploy applications on multiple environments either from a developer's local machine to an on-premises data center or to the cloud.

The most common examples of container engines are Docker and Kubernetes.

VMware Workstation Pro enables you to manage containers as well as virtual machines. With containers the **vctl** command, integrated into Workstation Pro, provides you with the ability to create, run, and manage containers from the command line and is the subject of this first chapter.

Getting Started with the vctl Command

Now we have brought you up to speed with what a container is and how it works; in this next section, we are going to get started with running containers in VMware Workstation Pro.

To run Kubernetes clusters and manage containers using Workstation Pro, the **vctl** command is used.

What Is the vctl Command

You can use the vctl command line utility in Workstation Pro to manage containers. In addition, the vctl command provides support for KIND (Kubernetes IN Docker) so that KIND can use vctl containers as nodes to run local Kubernetes clusters.

The vctl is a command line utility that is already bundled inside the Workstation Pro application and is supported only on host machines that are running Windows 10 1809 or later. If you are running Workstation

Pro on hosts that are running either a Linux-based operating system or Windows operating system version prior to Windows 10 1809, then these versions do not support the `vctl` command line, and therefore, it cannot be used.

We have just highlighted the fact that the required executable files are already included and bundled within in the Workstation Pro application, so the question is what they are and where would you find them.

To answer the latter part, you will find the files located in the `C:\Program Files (x86)\VMware\VMware Workstation` folder by default. If you choose to install Workstation Pro in a different folder location, then you will find them in the folder you created.

Next is what these files are called and what functionality do they provide. The three executables of the `vctl` command line utility are described below:

- **containerd.exe** – This is a runtime daemon that runs in the background on the host machine. The `containerd` daemon must be started first before you can run any container-related operation. To start it, you use the `vctl system start` command, and to stop it, use the `vctl system stop` command. We will cover these commands later in this chapter.
- **containerd-shim-crx-v2.exe** – When a new container is started, a new `containerd-shim-crx-v2` process is launched and acts as an adapter between the container in the CRX virtual machine and the `containerd` daemon.
- **bin/vctl.exe** – Is a command line utility that runs in the foreground and relays the user input to the `containerd` daemon.

Before you start running any of these commands, there are a couple of prerequisites you need to be aware of as we will discuss in the next section.

vctl Command Prerequisites

Although the vctl command is included with Workstation Pro and is available to run without the need to install any additional software or Workstation Pro features and is available from the standard command prompt windows or from a Windows PowerShell window, there are a few prerequisites you need to meet first as described below:

- VMware recommends that you use a modern solid-state drive (SSD) as system disk in the host machine to provide the required performance.
- The host operating system must be Windows 10 1809 or later.
- Before using the vctl command to run any operation on a container image or container, the container runtime must be started first.

The container runtime doesn't start automatically when the Workstation Pro application launches and does not stop automatically when you exit and close the Workstation Pro application. You must manually start and stop the vctl command which we will cover in the next section.

You can check as to whether the container runtime is running by following the steps described:

1. On the host machine that is running Workstation Pro, open a command prompt or a PowerShell session.
2. At the command prompt, type the following command: **vctl system info** and press Enter.

3. You will see the following screenshot with the red box highlighting the status of the container runtime which in this example shows that it is not currently running (stopped) as shown in Figure 1-2.

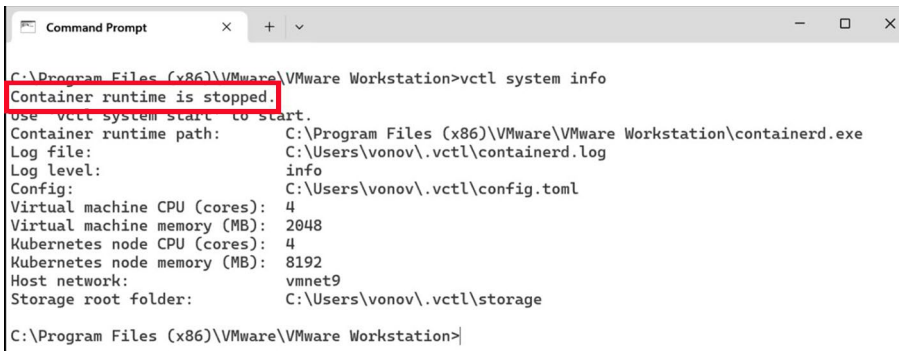


Figure 1-2. *Checking the Status of the Container Runtime*

4. If the container status showed as running, then you would be able to continue and manage the running containers using the vctl commands.

Having said all that, the first time you start the container runtime, a few more steps are required, which are automated, that basically download the virtual machine that acts as the container runtime for managing the Kubernetes nodes as well as the nodes themselves.

Setting Up and Managing a Container Runtime

Running the vctl command for the first time will download the required files and set up the container runtime environment before automatically starting it.

To do this, follow the steps described:

1. On the host machine that is running Workstation Pro, open a command prompt or a PowerShell session.
2. At the command prompt, type the following command: **vctl system start** and press Enter.
3. You will see the following as shown in Figure 1-3.

```

C:\Program Files (x86)\VMware\VMware Workstation>vctl system start
Downloading 3 files...
Downloading [crx.vmdk 95.17% kubect.exe 28.72% kind-windows-amd64 97.90%]
Finished kind-windows-amd64 100.00%
Downloading [crx.vmdk 98.20% kubect.exe 30.35%]
Finished crx.vmdk 100.00%
Downloading [kubect.exe 92.83%]
Finished kubect.exe 100.00%
3 files successfully downloaded.
Preparing storage...
Container storage has been prepared successfully under C:\Users\vonov\.vctl\storage
Preparing container network...
Container network has been prepared successfully using vmnet: vmnet9
Launching container runtime...
Container runtime has been started.

C:\Program Files (x86)\VMware\VMware Workstation>

```

Figure 1-3. *Starting the Container Runtime*

4. You will see that three files are downloaded:
 - **kind-windows-amd64** – Kubernetes IN Docker, local clusters for testing Kubernetes, is a tool for running local Kubernetes clusters using Docker container nodes. kind was primarily designed for testing Kubernetes itself.
 - **crx.vmdk** – This is the virtual disk for the CRX (Container Runtime Executive) virtual machine appliance used for the container host environment and is user for kind.

- **kubectl.exe** – kubectl is a command line tool that is used to run commands against Kubernetes clusters. It does this by authenticating with the master node of the cluster and then making API calls to perform the management actions.
5. These three files are saved in a folder called **bin** which in turn is created in a folder called **.vctl**. The folders are created the first time you run the vctl command.

The folders will all be created under the currently logged in user. In this example, the username of the currently logged in user is vonov, and so you will find the folders in the `c:\users\vonov\` folder as shown in Figure 1-4.

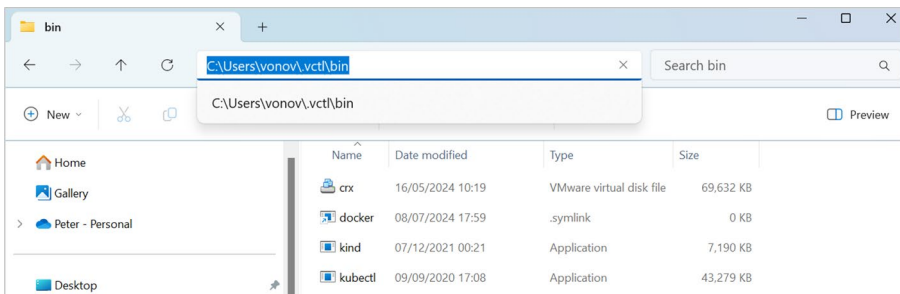


Figure 1-4. Downloaded Container Files

There are two other files that we are going to take a closer look at. Both files can be found in the **.vctl** folder. One file relates to the configuration of the Kubernetes node, and the other file relates to the CRX virtual machine appliance.

If you navigate to the `c:\users\\.vctl` folder, as shown in Figure 1-5, then you will see the files in question highlighted in red.

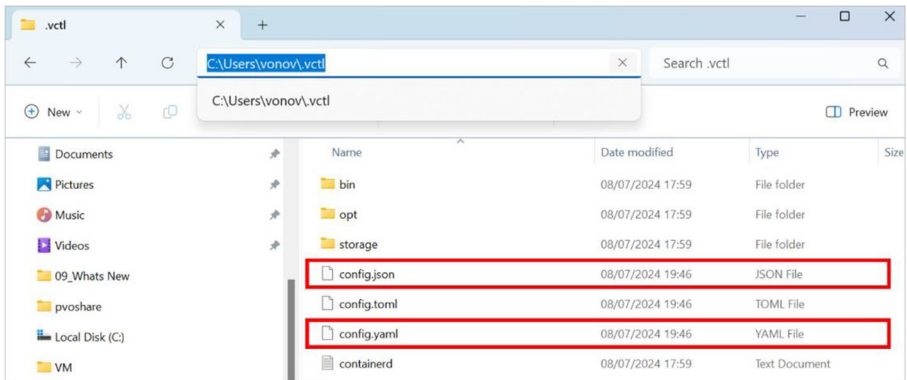


Figure 1-5. Container Configuration Files

If you open the **config.json** file using something like Notepad, you will see the following detail shown in Figure 1-6.

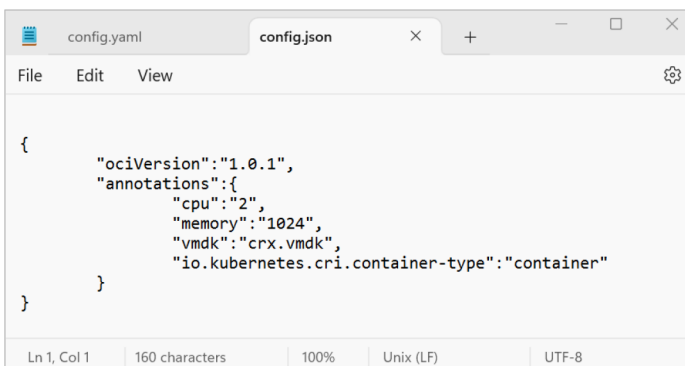
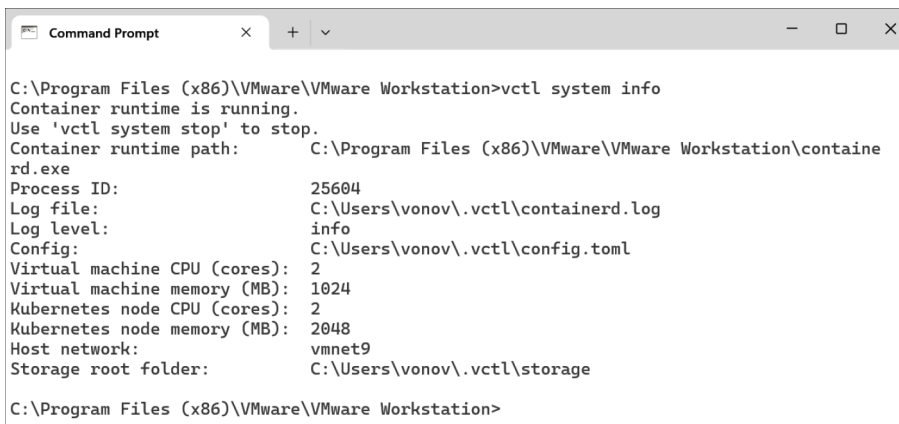


Figure 1-6. config.json File

As you can see from the config.json file, this virtual machine is configured with two CPUs and 1GB memory and uses a virtual disk file called crx.vmdk. But the question is, what is this virtual machine and why is it required when we are talking about running containers?

The name of the virtual disk file gives this away. CRX, as we have highlighted before, stands for Container Runtime Executive for ESXi. It is a virtual machine, as you can see from the configuration details, that is highly optimized to run a Linux kernel that in turn is highly optimized to enable you to run containers.

With the container runtime running, you will see the virtual machine and the Kubernetes node by running the **vtcl system info** command at the command prompt. This is shown in Figure 1-7.



```

C:\Program Files (x86)\VMware\VMware Workstation>vtcl system info
Container runtime is running.
Use 'vtcl system stop' to stop.
Container runtime path:      C:\Program Files (x86)\VMware\VMware Workstation\containe
rd.exe
Process ID:                  25604
Log file:                    C:\Users\vonov\.vtcl\containerd.log
Log level:                   info
Config:                      C:\Users\vonov\.vtcl\config.toml
Virtual machine CPU (cores): 2
Virtual machine memory (MB): 1024
Kubernetes node CPU (cores): 2
Kubernetes node memory (MB): 2048
Host network:                vmnet9
Storage root folder:         C:\Users\vonov\.vtcl\storage

C:\Program Files (x86)\VMware\VMware Workstation>

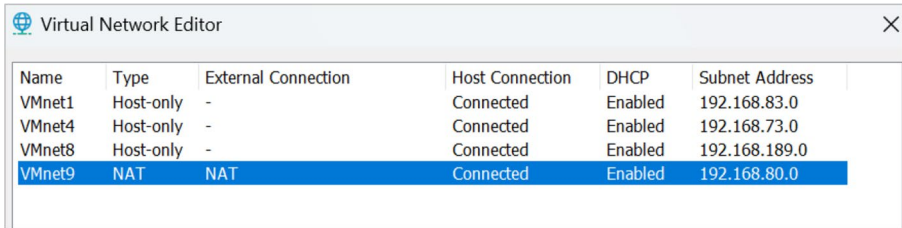
```

Figure 1-7. System Information and Status

On the subject of the container runtime, it will have been started as part of the vtcl system start command and once the files have been downloaded. The next time it starts, then it will just start without the need to download anything.

As the container service is now running, that means that vtcl-based KIND is now also ready. This means that kind will run local Kubernetes clusters by using vtcl containers as nodes and all Docker commands have been aliased to use vtcl so that Docker commands performed in the currently root opened window will be executed through vtcl.

We previously showed the configuration of the virtual machine and Kubernetes node in terms of CPU and memory, but you will also see that a new virtual network for containers, **vmnet9**, has been created. To show this, if you launch Workstation Pro and then open the **Virtual Network Editor**, you will see the new network as shown in Figure 1-8.



Name	Type	External Connection	Host Connection	DHCP	Subnet Address
VMnet1	Host-only	-	Connected	Enabled	192.168.83.0
VMnet4	Host-only	-	Connected	Enabled	192.168.73.0
VMnet8	Host-only	-	Connected	Enabled	192.168.189.0
VMnet9	NAT	NAT	Connected	Enabled	192.168.80.0

Figure 1-8. *New vmnet9 Network Created*

In the previous examples, we have just stuck with the default configuration values for CPU and memory for the virtual machine and the Kubernetes node, as well as the container storage volume size. In the next section, we are going to look at how to change or update the configuration.

Updating the CRX VM and Kubernetes Node Configuration

When it comes to the configuration of the virtual machine and the Kubernetes node, so far, we have stuck with the default configuration. However, depending on what you are planning on running, then there may be the need to change the configuration to add more memory or CPU resource.

These changes can be made when you start the container service, reconfiguring the default CPU and memory sizes available to the virtual machine and the Kubernetes node. First, let's look at increasing the resources on the virtual machine that hosts the Kubernetes node.

The command to do this would look something like the following:

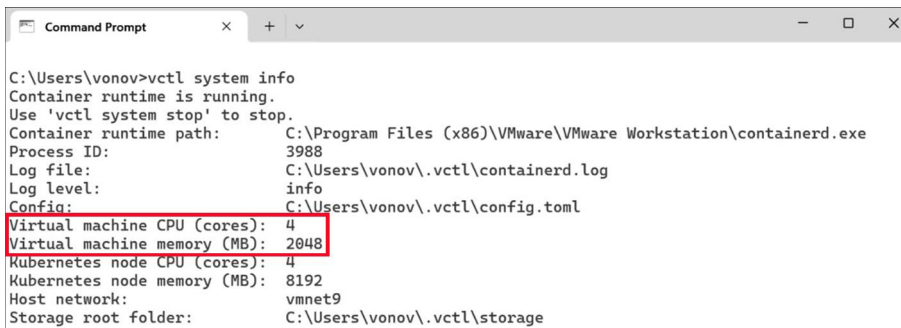
```
vctl system config --vm-cpus <CPU QTY> --vm-mem <MEMORY SIZE>
```

When running the command, replace **<CPU QTY>** with the number of CPUs you want to configure in the virtual machine, and then replace the **<MEMORY SIZE>** with the new memory size for the virtual machine.

In this example, we are going to change the virtual machine configuration to have four CPUs and 2GB of memory. The command to make that configuration change would look like the following:

```
vctl system config --vm-cpus 4 --vm-mem 2048
```

If you then run the **vctl system info** command, then you will see the following as shown in Figure 1-9.



```

Command Prompt
C:\Users\vonov>vctl system info
Container runtime is running.
Use 'vctl system stop' to stop.
Container runtime path:      C:\Program Files (x86)\VMware\VMware Workstation\containerd.exe
Process ID:                  3988
Log file:                   C:\Users\vonov\.vctl\containerd.log
Log level:                  info
Config:                     C:\Users\vonov\.vctl\config.toml
Virtual machine CPU (cores): 4
Virtual machine memory (MB): 2048
Kubernetes node CPU (cores): 4
Kubernetes node memory (MB): 8192
Host network:               vmnet9
Storage root folder:       C:\Users\vonov\.vctl\storage
  
```

Figure 1-9. Updated Virtual Machine Configuration

Now, let's look at increasing the resources for the Kubernetes node.

This command is almost identical to the previous command, but now we replace **vm** with **k8s**. Therefore, the command would look something like the following:

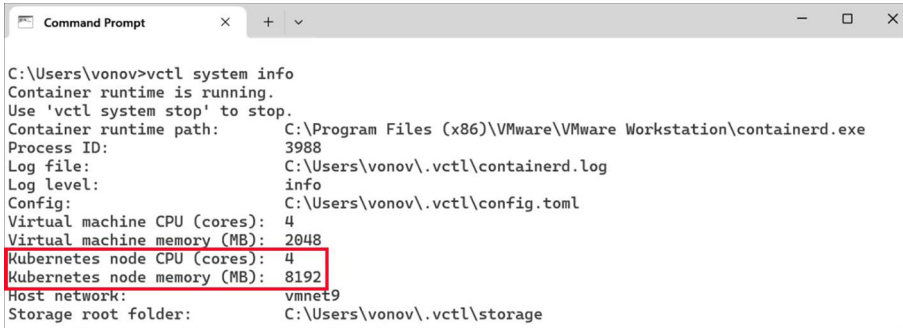
```
vctl system config --k8s-cpus <CPU QTY> --k8s-mem <MEMORY SIZE>
```


As with the previous command, replace **<CPU QTY>** with the new number of CPUs you want to configure in the Kubernetes node, and then replace the **<MEMORY SIZE>** with the new memory size for the Kubernetes node.

In this example, we are going to change the Kubernetes node configuration to have four CPUs and 8GB of memory. The command to make that configuration change would look like the following:

```
vctl system config --k8s-cpus 4 --k8s-mem 8192
```

If you then run the `vctl system info` command, then you will see the following as shown in Figure 1-10.



```

Command Prompt
C:\Users\vonov>vctl system info
Container runtime is running.
Use 'vctl system stop' to stop.
Container runtime path: C:\Program Files (x86)\VMware\VMware Workstation\containerd.exe
Process ID: 3988
Log file: C:\Users\vonov\.vctl\containerd.log
Log level: info
Config: C:\Users\vonov\.vctl\config.toml
Virtual machine CPU (cores): 4
Virtual machine memory (MB): 2048
Kubernetes node CPU (cores): 4
Kubernetes node memory (MB): 8192
Host network: vmnet9
Storage root folder: C:\Users\vonov\.vctl\storage
  
```

Figure 1-10. Updated Kubernetes Node Configuration

As you have updated the configuration of both the virtual machine and the Kubernetes node, then the corresponding configuration files will also have been updated, therefore making these changes persistent.

If you open the **config.yaml** and the **config.json** files, then you will see this updated configuration within the respective configuration files.

Now that we have downloaded the required files and configured and started the container service, we can now turn our attention to the commands that can be used in building and managing containers.

vctl Commands

We have already looked at some of the vctl commands in the previous section. Those commands were used to configure the CRX virtual machine and the Kubernetes node.

The next set of commands we are going to cover is for building, controlling, and managing the containers themselves. We will cover these commands in the following sections.

The build Command

The build command enables you to build a container image from a Dockerfile. To run the build command, you would use the following syntax:

```
vctl build <COMMAND_OPTIONS> <PATH>
```

When running the command, replace the **<COMMAND_OPTIONS>** field with the build option you require, which we will discuss next, and replace the **<PATH>** field with the path to the Dockerfile from which you want to build your image.

build Command Options

The build command has several options that can be specified when running the command that allows you to configure how the container image is built. The command options are as follows:

--builder-mem <string> - Configures a limit for the amount of memory that is available to the container. You can enter MB or GB). By default, the container is built with 4GB (4g). Replace the **<string>** field in the command with the amount of memory you want to configure.

-c, --credential <string> - Enables you to specify the path to the file that is used to store the private registry authentication credentials. Replace the **<string>** field in the command with the path.

-f, --file <string> - Enables you to configure the path to the target Dockerfile that the container will be built from. By default, the path is set to `PATH\Dockerfile`. Replace the **<string>** field in the command with the name of the Dockerfile.

-h, --help - Displays the help screen. In this case, by using the build command, the help shown will be specific to the build command.

--kind-load - Enables you to load the container image to a local kind cluster.

--no-local-cache - Configures the container to not use local storage as cache for the base image.

-t, --tag <string> - Enables you to specify the name of the container image that gets built. Replace the **<string>** field in the command with the tag you want to use.

With each of the command options described above, where you see the **<string>** field requires you to input a string value such as entering a name or the amount of memory.

Next, we are going to look at the completion command.

The completion Command

With the completion command, you can output the shell completion code for reporting on whether the shell loading has completed successfully. You have the choice of the shell that you output the completion code to with the command supporting Bash, Zsh, Fish, and PowerShell.

As a prerequisite for this command to work, you will need to have your chosen shell application installed and the completions feature enabled where appropriate. In the next sections, we are going to look at the syntax for each of the supported shell options.

Bash

If you want to output the completion code to Bash, then you would run the completion command for each session using the following syntax:

```
vctl completion bash > /usr/local/etc/bash_completion.d/vctl
```

Zsh

If you want to output the completion code to Zsh, then you would run the completion command for each Zsh shell session using the following syntax:

```
vctl completion zsh > "${fpath[1]}/_vctl"
```

Note For this command to take effect, then you will need to start a new shell.

Fish

If you want to output the completion code to Fish, then there are two options. The first option is to output the completion code to the current shell. To do this, you would use the following syntax:

```
vctl completion fish | source
```

The second option is to load the completion code for each session. The command syntax for this option is as follows:

```
vctl completion fish > ~/.config/fish/completions/vctl.fish
```

Finally, we are going to look at the PowerShell option.

PowerShell

Finally, there is the option to output the completion code to PowerShell.

Again, there are two options with the completion command when used with PowerShell. The first option executes the completion command once using the following command syntax:

```
PS> vctl completion powershell | Out-String | Invoke-Expression
```

The other option is to load completions for every new session, using the following command syntax:

```
PS> vctl completion powershell > vctl.ps1
```

In the next section, we are going to look at the **create** command.

The create Command

With the **create** command, you can create a new container from an existing container image.

To run the build command, you would use the following syntax:

```
vctl create <OPTIONS> <IMAGE> <COMMAND> <ARGUMENTS>
```

When running the command, replace the **<OPTIONS>** field with the build options you require, which we will discuss next. Then replace the **<IMAGE>** field with the path to the image file from which you want to create a new container image from. Then you can add commands in the **<COMMAND>** field and any arguments for those commands in the **<ARGUMENTS>** field.

create Command Options

The create command has several options that can be specified when running the command that allows you to configure how the container image it creates. The command options are as follows:

- **--entrypoint <string>** – Enables you to override the default entry point of the container image
- **-e, --env <strings>** – Enables you to set environment variables in the container
- **--hostname <string>** – Enables you to set a hostname for the container
- **-i, --interactive** – Enables you to keep the STDIN (standard input) open even if not attached
- **-l, --label <strings>** – Enables you to configure additional labels on the container
- **-n, --name <string>** – Enables you to assign a name to the container
- **-r, --privileged** – Enables you to run the container with extended privileges