D085 Tips and Examples

Hello!

Thank you for taking the time to review this resource! This document was created to address the common questions and concerns encountered when working on the performance assessment. Your assessment will be reviewed and evaluated by another team and not by your instructor. Unlike Objective Assessments which have a maximum number of 4 attempts, performance assessments (which this course uses) have **no maximum on attempts**, so this may relieve some of the stress when learning the material. If you are [catastrophizing,](https://psychcentral.com/lib/what-is-catastrophizing/) overthinking, running into [analysis paralysis](https://en.wikipedia.org/wiki/Analysis_paralysis), or get stuck for longer than 10 minutes, please reach out and ask for help. We are always happy to help!

You are not limited to or required to use the course learning resources to complete the Performance Assessment. You are free to use any resources as you see fit to complete the performance assessment, and we recommend working on it as you go through the course so you know what you need to learn to complete it. In fact, it is very common to research how to perform a task using google, stack overflow, and vendor documentation ([learn.chef.io](https://learn.chef.io/modules#/), [docs.chef.io](https://docs.chef.io/) and its future home at [chef.sh](https://www.chef.sh/docs/chef-workstation/getting-started/) in this case) when you are developing a solution as an engineer. Learning how to learn is a key skill to being successful in many IT and engineering jobs! To help with learning the material, we highly recommend embracing a [growth mindset](https://www.brainpickings.org/2014/01/29/carol-dweck-mindset/) (the audiobook can be found on [youtube](https://www.youtube.com/results?search_query=mindset+the+new+psychology+of+success)).

Below are tips that you may find helpful when working on the Performance Assessment:

# Table of Contents

[Table of Contents](#_Toc98257215)

[Tips and General Info](#_Toc98257216)

[Getting Started - How to setup your environment](#_Toc98257217)

[A. Introduction of Solution](#_Toc98257218)

[B. Visual Representation](#_Toc98257219)

[C. Automation Script](#_Toc98257220)

[D. Diagnostic Report](#_Toc98257221)

[Diagnostic Report Example](#_Toc98257222)

[How to get the Data](#_Toc98257223)

[E. Web Sources](#_Toc98257224)

[F. Sources](#_Toc98257225)

[G. Professional Communication](#_Toc98257226)

[How to submit your solution](#_Toc98257227)

[Errors and fixes](#_Toc98257228)

[Error parsing kitchen.yml](#_Toc98257229)

[Could not load the chef provisioner](#_Toc98257230)

[Unable to ping from VM1 to VM2](#_Toc98257231)

[**Stderr: VBoxManage.exe: error: Not in a hypervisor partition (HVP=0) (VERR\_NEM\_NOT\_AVAILABLE).**](#_Toc98257232)

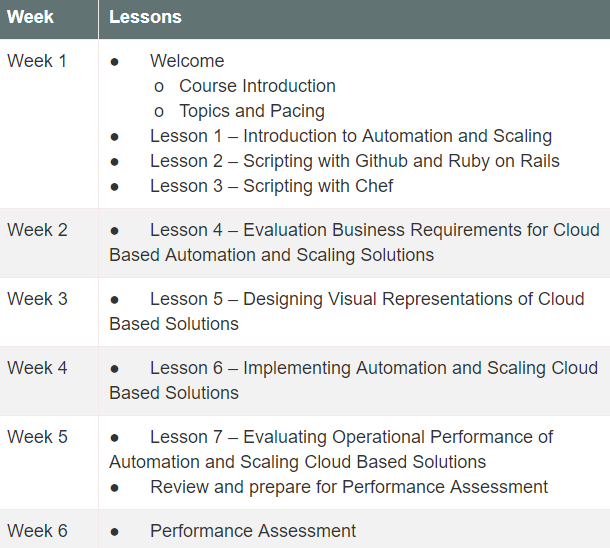
[Packer errors using windows server 2012](#_Toc98257233)

[Chef Workstation cannot execute without accepting the license](#_Toc98257234)

[How to get help](#_Toc98257235)

# Tips and General Info

Below are tips and general info about the course:

* We recommend jumping in and start working on the Performance Assessment as you go through the course. Use the tips and examples found in this document for guidance on the Performance Assessment. A great intro to installing and setting up chef, test kitchen and vagrant can be found[here](https://learn.chef.io/courses/course-v1:chef+LocalDev101+Perpetual/courseware/f2d0e26dc6f941358a0e7ea2be84a26a/92e258cafac5461794e0d466384ae591/1?activate_block_id=block-v1%3Achef%2BLocalDev101%2BPerpetual%2Btype%40vertical%2Bblock%403232c5b376fc4662a807082722f6ee5e). The [How to setup your environment](#_3r0ug5lgv2wc) section has videos that walk through this tutorial and the setup process.
* After getting set up, it may be easier to work on section C followed by section D before working on sections A and B. Having your environment setup and data collected will make the essay portion and visual representation portion easier as you will have something to reference when working on these sections.
* You may use AWS or Azure to follow along and to complete the performance assessment. However, An **AWS environment for the course is not needed** and everything can be done locally on your computer using docker or vagrant.
* We recommend that you **avoid writing solutions from scratch** as there is a good chance someone has already solved the problem. Github, googling, and stack overflow are good resources. With that being said, some modifications to code will be required, ensure that you understand the code and solution that is used, and avoid questionable sources which may lead to bad code.
* If you would like to use AWS, you can sign up for the [free tier](https://aws.amazon.com/free/). Keep in mind that it requires that you place a credit or debit card on file to use it.
* The average time taken to complete the course is **10 days**. You can use the **five day challenge** below if you really want to push yourself:
  + **Day 1**: get environment setup using [how to set up environment](#_3r0ug5lgv2wc) section
  + **Day 2**: Complete Section C - Automation Script & D - Diagnostic Report
  + **Day 3**: Complete Section A - Introduction of Solution
  + **Day 4**: Complete Section B - Visual Representation
  + **Day 5**: Review, cite sources ([Sections E](#_tb4d5b23jgb0) and [F](#_bs4wqwh9zw8w)) & submit solution for evaluation!
* You can also use the pacing guide below to pace your completion of the course. With that being said, you are free to proceed with the course as you see fit :   
  

# Getting Started - How to setup your environment

The following videos walk through setting up the environment. Click the links or the pictures below to view each video:

NOTE: you can use the following github repository if the one in the video does not exist:

<https://github.com/2f9743hfjkdw9/learn_chef_httpd.git>

|  |  |
| --- | --- |
| [D085 setup pt1 - installing components](https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=8af7d2a5-b9b7-4234-a5d7-ab430039fc3c)  ￼ | [D085 setup pt2 - setting up test kitchen with virtualbox and vagrant](https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=be04b2af-e681-434b-b7c4-ab43010be3b5)  ￼ |

**NOTE**: Make sure you using [visual studio code](https://code.visualstudio.com/) or powershell when connecting to instances. Do not use powershell ISE, ssh does not work and it throws false errors. Also make sure [hyper-v is disabled](http://rizwanansari.net/run-hyper-v-and-virtualbox-on-the-same-machine/) if you are using vagrant and virtualbox.

Once this is done, you should have one instance/server running ubuntu or centos.  All you have to do at this point is modify the [**kitchen.yml**](https://docs.chef.io/config_yml_kitchen.html) to create a node/server for each of the servers from the PA.   Below is an example of what this will look like.  The script below creates two VMs called test1 and test2.  You can build on this example to create the 8 VMs needed.  You can give them any name you wish.

**NOTE**:  We have seen a high number of returned submissions for plagiarism in the automation script (section C).  Please make sure that your submission is unique and not copied word for word from another resource.  To do this, make sure you use unique names for your VMs and that the comments in the script are not copied word for word from the tips doc or another resource used.  You can give the VMs any name you wish, and they can be in any order, but please make sure the names are unique.

|  |
| --- |
| driver:  name: vagrant  network:  - ["public\_network"] # creates second interface to allow VMs to reach each other  customize:  memory: 512 # sets the amount of memory for each VM to 512 MB  provisioner:  name: chef\_zero  product\_name: chef  product\_version: 14.12.9  # verifier specifies which application to use when running automated tests.  verifier:  name: inspec  # platforms specifies the target operating systems. We're targeting just one CentOS 7.  platforms:  - name: centos-7  suites:  - name: test1  driver:  vm\_hostname: test1.SparkIT-Games.com  run\_list:  - recipe[learn\_chef\_httpd::default]  attributes:  - name: test2  driver:  vm\_hostname: test2.SparkIT-Games.com  run\_list:  - recipe[learn\_chef\_httpd::default]  attributes: |

An explanation of the parameters in the kitchen.yml can be found in the [kitchen.yml page from the chef.io documentation here](https://docs.chef.io/config_yml_kitchen.html):

|  |
| --- |
| * **Driver**: driver specifies the software that manages the machine. In the example above, we're using the Vagrant Test Kitchen driver. A list of other popular drivers that can be used are found [here](https://docs.chef.io/config_yml_kitchen.html#drivers). The Vagrant driver works with VirtualBox by default.   + **Network**: Used to specify an array of network customizations to be applied to the virtual machine. In the example above, we are using - **["public\_network"]** to tell vagrant to create a second public interface for each VM. This will be used to allow connectivity between VMs and to other devices on our network.   + **Customize**: We use **memory: 512** to set the amount of memory for each VM to **512 MB**. * **Provisioner**: provisioner specifies how to run Chef. We use chef\_zero because it enables you to mimic a Chef server environment on your local machine. This allows us to work with node attributes and other Chef server features. * **Verifier**: specifies which application to use when running automated tests. We are using inspec. * **Suites**: suites specify what we want to apply to the virtual environment. You can have more than one suite. Here we define one for each of the servers for the PA.   + **Name:** This defines the name we want to use for each instance. We make one for each server to be used for the PA.   + **vm\_hostname**: the hostname to be used for each instance.   + **Run\_list**: This defines which recipes to run and the order to run them on the instance. The recipes are stored in the recipes folder inside your cookbook (**learn\_chef\_httpd**). In this example, we are using the **default** recipe from the [getting started with test kitchen module](https://learn.chef.io/modules/local-development/rhel/virtualbox/apply-a-cookbook#/) from learn.chef.io. |

You can leave the **default.rb** recipe inside the recipes folder as is. Each of our instances will use it and will have apache running once we are done. Yours may look different than the one below, that is ok. You can also just use the one below as well:

|  |
| --- |
| #  # Cookbook Name:: learn\_chef\_httpd  # Recipe:: default  #  # Copyright (c) 2016 The Authors, All Rights Reserved.    # references:  # how to apply recipies - https://learn.chef.io/modules/local-development/rhel/virtualbox/apply-a-cookbook#/  # how to create recipies - https://learn.chef.io/modules/learn-the-basics/rhel/virtualbox/configure-a-package-and-service#/  # chef marketplace (where to get recipies) - https://supermarket.chef.io/  # about recipies - https://docs.chef.io/recipes.html    # install apache  package 'httpd'    # run apache web server when instance starts  service 'httpd' do  action [:enable, :start]  end    # creates a default homepage with "hello world"  file '/var/www/html/index.html' do  content '<html>  <body>  <h1>hello world</h1>  </body>  </html>'  end |

Run **kitchen list** and you will see the new instances ready to be created:

|  |
| --- |
|  |

Run **kitchen create** and it will create the instances:

|  |
| --- |
| … (Truncated)… |

Then run **kitchen converge** to install the chef client tools and the recipes on each VM:

|  |
| --- |
| … (Truncated)… |

You are now ready to start filling out the Diagnostic Report Template! Copy the **kitchen.yml** used above and paste it in the part[**C – Automation Script**](#_p03hei523bag)section of your diagnostic report document. Then follow the instructions and examples in [**Section D – Diagnostic Report**](#_wkilaf5b5sxj) for guidance on how to complete that section.

# A. Introduction of Solution

This section should describe the proposed solution, including the purpose, goals and objectives, scope, and a brief description of how the solution functions. It should be a page in length. Below is an example of a way to approach this section where you dedicate a paragraph to each of the following (you do not have to do it this way, this is only a suggestion):

* Introduction of solution
* Purpose of the project
* Goals and objectives
* Scope (describe the environment and systems used)
* Description of how your solution functions

Below is an example:

|  |
| --- |
| **Purpose**  The purpose of this solution is to provide a flexible and scalable platform that will scale  with the needs of the SparkIT-Game Company’s upcoming release of a new version of its  flagship game.  **Goals and Objectives**  The goals and objectives of this solution are to meet the scaling needs of SparkIT-Game  Company ahead of the upcoming PAX West conference. This includes scaling the clusters to  reliably handle up to eight million concurrent users while also featuring dynamic scaling  capabilities that should scale the clusters back in during non-peak times to save money.  Additionally, SparkIT-Game Company would like to utilize modern tools and frameworks  in order to best leverage the latest capabilities as well as attract top talent to join the team. To  meet this goal, this solution will run leverage the AWS EKS hosted Kubernetes³ solution as well as follow Open Container Initiative specifications for the deployment of containerized  applications. To meet this goal, the solution will leverage Chef to deploy the CRI-O container  runtime purpose built for Kubernetes as the container host platform for the clusters.²  The CRI-O container host clusters running under the orchestration of an EKS  Kubernetes control plane will host containerized instances of all required applications in each  cluster in order to maintain redundancy.  **Scope**  New clusters will be created when a CloudWatch metric that tracks the  RequestCountPerTarget reaches the threshold of 1000 requests that individual application  targets can adequately handle.There are 40 targets within a cluster, totaling in 40,000  concurrent users per cluster. In order to be proactive, a new cluster will begin spinning up when there are 30,000 backend requests going to a cluster currently in service. Cooperative play will scale at 20,000 requests rather than 30,000 requests, as cooperative play clusters cannot handle as many requests as micropayment, core, and backend clusters. When the number of requests are 40,000 more than the total ​possible​ requests for backend, core, and micropayment clusters and 20,000 for cooperative play clusters, the number of clusters will scale back in to save money. When new clusters are created or destroyed, an AWS SNS notification will be sent to helpdesk@SparkIT\_game.com.  The AWS EC2 Autoscaling Group will also remove clusters if CloudWatch logs indicate a  high number of errors is occurring in a particular cluster. Alerts about elevated errors will also be emailed to helpdesk@SparktIT\_game.com.  **Functionality**  There are a few tools and platforms used to deploy and operate the solution. First, the  EKS control plane, EC2 load balancers, CloudWatch metrics and alarms, EC2 AutoScaling  Group, and DNS records will be deployed with Terraform. Next, when new servers are spun up in the AutoScaling Group, Chef cookbooks will run to install the CRI-O platform for container hosting and associate the hosts with the Kubernetes control plane.  From there, the platform should add additional clusters and scale out as needed and  scale in to save resources when utilization is lower. The help desk will be notified when errors  and/or scaling events occur |

# B. Visual Representation

Create a visual representation of your solution (e.g., storyboard, flowchart, UML diagram, GUI) that illustrates how the system process or workflow aligns with and supports the business requirements for SparkIT-Game Company. Include the executable tasks and decision points in your representation. You can use Visio, Draw.io, or powerpoint to do this. Please make sure to include **explicit numbers or conditions** that will be used for the decision points. For example: When CPU utilization is >50% for a cluster a second cluster is deployed.

The visual representation does not have to include load balancer as long as it makes it obvious when the second cluster will be created (ex:  once the game reaches 30,000 users).  Below is an example of what your visual representation could look like.  Take notice how each example labels each decision point with a number.  In addition to cluster 1 shown, please make sure that your diagram also has cluster 2 and the co-op cluster.  Please also make sure you have a decision point that shows when the number of servers will be reduced.

|  |
| --- |
|  |

Here is another example:

|  |
| --- |
| 1. Automation VM running Vagrant and Kitchen consists the following VM devices:  •         Core VM  •         Web Front End  •         Web Back End  •         Micropayment  •         DataBase  Each VM can serve 40,000 users.  4. When cpu usage is greater than 60% for the CG2 Clulter 1, then CG2 Cluster 2 will be deployed automatically.  10. Reporting - Alerts like activation and deactivation of each new cluster will send a message to the help desk ticketing system via helpdesk@SparkIT-Game.com/ |

# C. Automation Script

The script created will be used to create the 8 VMs used in part D.  Below is an example of what this will look like.  The script below creates two VMs called test1 and test2.  You can build on this example to create the 8 VMs needed.

**NOTE**:  We have seen a high number of returned submissions for plagiarism in the automation script (section C).  **Please make sure that your submission is unique and not copied word for word from another resource**.  To do this, make sure you use unique names for your VMs and that the comments in the script are not copied word for word from the tips doc or another resource used.  You can give the VMs any name you wish, and they can be in any order, but please make sure the names are unique. **Please also copy and paste the script in your diagnostic report document (should be text, not an image) and include a screenshot of the script executing by taking a screenshot of the result of running kitchen create** (the same command you’ll use on section D row 1)**.**

|  |
| --- |
| driver:  name: vagrant  network:  - ["public\_network"] # creates second interface to allow VMs to reach each other  customize:  memory: 512 # sets the amount of memory for each VM to 512 MB  provisioner:  name: chef\_zero  product\_name: chef  product\_version: 14.12.9  # verifier specifies which application to use when running automated tests.  verifier:  name: inspec  # platforms specifies the target operating systems. We're targeting just one CentOS 7.  platforms:  - name: centos-7  suites:  - name: test1  driver:  vm\_hostname: test1.SparkIT-Games.com  run\_list:  - recipe[learn\_chef\_httpd::default]  attributes:  - name: test2  driver:  vm\_hostname: test2.SparkIT-Games.com  run\_list:  - recipe[learn\_chef\_httpd::default]  attributes: |

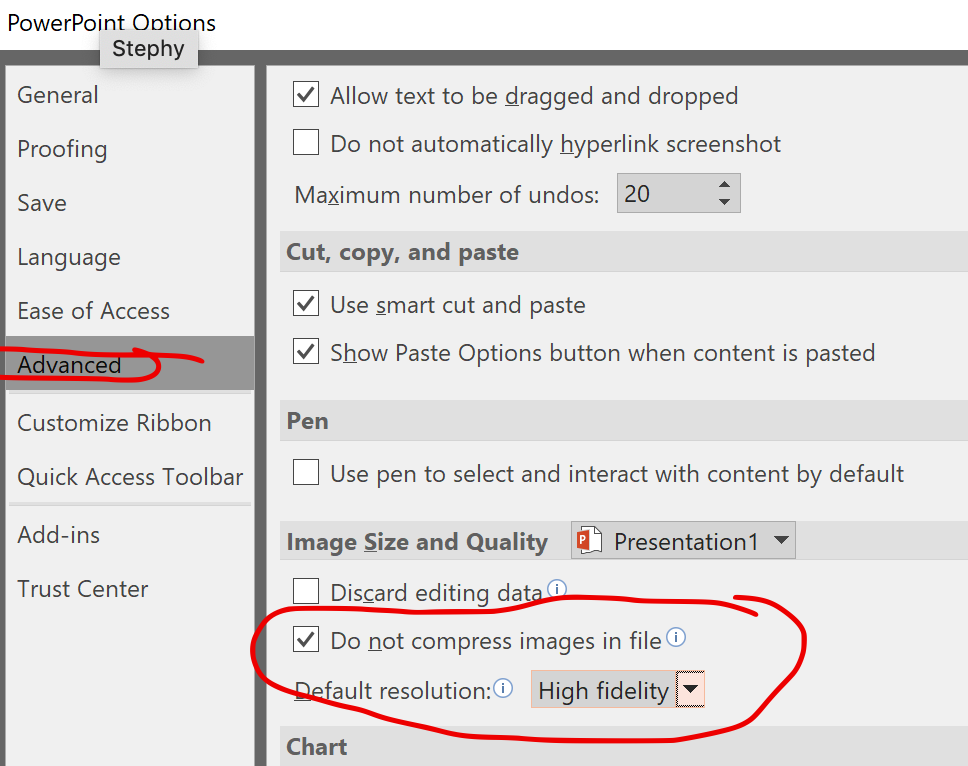
An explanation of the parameters in the kitchen.yml can be found in the [kitchen.yml page from the chef.io documentation here](https://docs.chef.io/config_yml_kitchen.html).

* **Driver**: driver specifies the software that manages the machine. In the example above, we're using the Vagrant Test Kitchen driver. A list of other popular drivers that can be used are found [here](https://docs.chef.io/config_yml_kitchen.html#drivers). The Vagrant driver works with VirtualBox by default.
  + **Network**: Used to specify an array of network customizations to be applied to the virtual machine. In the example above, we are using - **["public\_network"]** to tell vagrant to create a second public interface for each VM. This will be used to allow connectivity between VMs and to other devices on our network.
  + **Customize**: We use **memory: 512** to set the amount of memory for each VM to **512 MB**.
* **Provisioner**: provisioner specifies how to run Chef. We use chef\_zero because it enables you to mimic a Chef server environment on your local machine. This allows us to work with node attributes and other Chef server features.
* **Verifier**: specifies which application to use when running automated tests. We are using inspec.
* **Suites**: suites specifies what we want to apply to the virtual environment. You can have more than one suite. Here we define one for each of the servers for the PA.
  + **Name:** This defines the name we want to use for each instance. We make one for each server to be used for the PA.
  + **vm\_hostname**: the hostname to be used for each instance.
  + **Run\_list**: This defines which recipes to run and the order to run them on the instance. The recipes are stored in the recipes folder inside your cookbook (**learn\_chef\_httpd**). In this example, we are using the **default** recipe from the [getting started with test kitchen module](https://learn.chef.io/modules/local-development/rhel/virtualbox/apply-a-cookbook#/) from learn.chef.io.

Contrary to what the BDBR document states, the “scripts” referenced in that document and seen below are just the **names** of each of the servers that you can create and use for the PA. They are not scripts and are not provided as part of the support documents.

|  |
| --- |
| The following scripts must use preconfigured template images in protected stores:  • //cloud\_images/CGSP-2-core (for core services)  • CG2-web-front (for front-end web images)  • CG2-web-back (for back-end web services)  • //cloud\_images/CG-db (for data services)  • //cloud\_images/CG-2-Micropayment (for transaction services)  • //cloud\_images/CG-2-CoopPlayCore  • //cloud\_images/CG-2-CoopPlay-gw  • //cloud\_images/CoopPlay-env for the cooperative play module |

# D. Diagnostic Report

* You **do not** have to gather this data from each node in your cluster, one will do.
* Rows **1, 2, 5, and 15** can be obtained using **test kitchen commands** as shown in the diagnostic report example below.
* Most of the performance metrics can be gathered with standard tools (ping, tracert/traceroute, iostat, etc.). **Please see below in the diagnostic report table and after that for examples.** You can use  [**kitchen exec -c ‘COMMAND’**](https://docs.chef.io/ctl_kitchen.html#kitchen-exec) or [**vagrant ssh <instanceid>**](https://www.vagrantup.com/docs/cli/ssh.html) as shown in the [how to get the data](#_e5xfkfpa5pd) section to run these commands.
* The report should be completed as follows for each row:
  + **Data and Results –** Provide the data received for each of the requested metrics from the Data Description column.
  + **Script or command Used to Extract data –** You **do not have to use a script** to gather the data for each row. Most of the data requested can be collected using the commands and tools included on most systems (ping, tracert/traceroute, iostat, etc.). If you create or use a script to gather the data in this row, then please state the name and include it in your submission. Please make sure you include the **full command** you ran to obtain the data for each row and not just the name. For example: **ping <ip> -p <port>**, **dd if=<path>, of=<path>**, etc) .
  + **Screenshot of Result of Script** – take a screenshot showing the method and result you used to get the data in columns 3 & 4, and insert the screenshot into the table. For example, if you used the top command to get the load (1.5, 2.2, etc.) the screenshot should show the top command displaying the load. **Its ok if the screenshot is small, we will zoom in to view the result. Just make sure that the output in the screenshot can be read.**
* You may also need to disable image compression in microsoft word to avoid blurry images.
  + Click File > Options. In the Options box, click Advanced. Under **Image** Size and Quality, select the Do not **compress images** in file check box.  
    

## Diagnostic Report Example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data Description** | **Optimal Range** | **Data and Results** | **Script or command Used to Extract Data** | **Screenshot of Result of Script (zoom in to view image or follow link)** |
| Time to scale from 1 cluster to 200 clusters  (8 million users expected at peak after PAX West) based on 40K users per cluster (subject to change based on load testing)  **You are not expected to create 200 clusters. You only need the nodes created from the kitchen.yml in part C. This will be your cluster.** | 15–30 minutes for each cluster | Give the time to create a cluster. You can get this time by running kitchen create. Use that time to estimate the time to deploy 200 clusters. For example:    **For one cluster = 9m26s**  **200 clusters = 9m26s \* 200 = 1886 minutes (31 hours)** | Kitchen create, kitchen converge  OR  Kitchen test | [Screenshot](https://drive.google.com/file/d/1E8OjqjfnIyUVLkXuE3tqdXVBKBP6xcPs/view?usp=sharing)1  [screenshot2](https://drive.google.com/file/d/1x42mRhbozzCEbthhoLt7eFPfMuJ8MGU0/view?usp=sharing) |
| Time to register a cluster and then quench connections to the load balancer, taking the cluster off-line (start-up, operation, shutdown) | 1 minute per connection quench, start of cluster launch, and part of time to scale cluster, can be tracked separately as a quench | Give the time taken to destroy the cluster. For example:  **2m14s** | kitchen destroy. | [Screenshot](https://drive.google.com/file/d/17veaYjtZuMgALxZdZrziw9QNx8332fwd/view?usp=sharing) |
| Peak load averages per system at 20K, 30K, and 40K users per game cluster | 60% of CPU triggers new cluster launch; if reaching core load at 20K users, launch new cluster on 60% CPU loads | Give the current system load for any instance. For example:    **load average: 0.00, 0.01, 0.05**    Ex2: threshold set (if using AWS) | Top    AWS load balancing - [ELB](https://aws.amazon.com/elasticloadbalancing/)(based on resource usage) | [Screenshot](https://drive.google.com/file/d/1y5vaPU5yV8GgTu7t677fE5yX0JD3aUGU/view?usp=sharing)    Screenshot from AWS showing balancing setup used |
| Write times to the diagnostic data drive | <30 milliseconds | Time to write data to the drive. You just need to show disk write performance. For example:  **1073741824 bytes (1.1 GB) copied, 6.84678 s, 157 MB/s** | [dd](https://www.thomas-krenn.com/en/wiki/Linux_I/O_Performance_Tests_using_dd) (use to write data to the drive and record the time it takes)  [Winsat](https://superuser.com/questions/130143/how-to-measure-disk-performance-under-windows) (if you are using windows) | [Screenshot](https://drive.google.com/file/d/1UxMxgb307WiIyspzbHLiEfDrT3HC9gIF/view?usp=sharing) |
| Pull time from the game instances (1 core, 1 web front end, 1 web back end, 1 database, and 1 micropayment server) and initialization time | Part of cluster launch 15–30 minutes | Provide time to start cluster (this is similar to row 1). For example:  **9m26s** | Kitchen create, kitchen converge, | [Screenshot](https://drive.google.com/file/d/1E8OjqjfnIyUVLkXuE3tqdXVBKBP6xcPs/view?usp=sharing)1  [screenshot2](https://drive.google.com/file/d/1x42mRhbozzCEbthhoLt7eFPfMuJ8MGU0/view?usp=sharing) |
| Average messaging service (queue) time | <1 minute in queue | Give messaging service time | If you are using vagrant:  **time curl -X POST -H "Content-Type: application/json"**  **-d '{"userId": 48593, "title": "messaging service test", "body": "Testing messaging service queue time."}'     https://jsonplaceholder.typicode.com/posts**  If you are using AWS, you can [use sqs](https://aws.amazon.com/getting-started/tutorials/send-messages-distributed-applications/).  If you are using Azure, you can use [storage queues](https://www.tutorialspoint.com/microsoft_azure/microsoft_azure_queues.htm) [with Azure storage explorer](https://www.tutorialspoint.com/microsoft_azure/microsoft_azure_queues.htm).  Use [zeromq](https://learning-0mq-with-pyzmq.readthedocs.io/en/latest/pyzmq/patterns/client_server.html) | [screenshot](https://westerngovernorsuniversity-my.sharepoint.com/:i:/g/personal/michael_mckinney_wgu_edu/EbjShTHTW5BHuwL4HDnWI5cBEIb9hkcGypfj6ycEFQFAsg) |
| Average latency for the micropayment server | <30 milliseconds | Do a ping from micropayment instance to any web site  **rtt min/avg/max/mdev = 15.011/16.016/18.375/1.202 ms** | Ping google.com | [screenshot](https://drive.google.com/file/d/1cIsXM3TauPpyhdk4yb4b2AFwq2ad8JH_/view?usp=sharing) |
| Average latency of each cluster | <30 milliseconds | Ping localhost or name of instance. For example:  **rtt min/avg/max/mdev = 0.009/0.022/0.031/0.008 ms** | Ping localhost (from any node) | [screenshot](https://drive.google.com/file/d/1jvbixu2TgUKzsExinonzEu7ujoOPAYW0/view?usp=sharing) |
| Network data in and out for each cluster | <1 second | Record bandwidth or ping time to any instance. For example:  **rtt min/avg/max/mdev = 14.546/15.162/16.290/0.693 ms** | Ping, scp, or nload (from any node) | [Screenshot](https://drive.google.com/file/d/1VWqe06YnAZmplUE8Et5VtXJ_IGAGekxe/view?usp=sharing) |
| Overall CPU utilization of the environment for each cluster | Not >60% | Give the CPU load for one of the nodes. For example:  **load average: 0.00, 0.01, 0.05** | top | [Screenshot](https://drive.google.com/file/d/1y5vaPU5yV8GgTu7t677fE5yX0JD3aUGU/view?usp=sharing) |
| Diagnostic data able to be written by the automation to the correct cloud bucket storage space | Show read/write times <1 second | Give the time to write data to cloud bucket storage space (amazon S3, [azure blob storage](https://www.tutorialspoint.com/microsoft_azure/microsoft_azure_blobs.htm) ). For example:  **1073741824 bytes (1.1 GB) copied, 6.84678 s, 157 MB/s** | [dd](https://www.thomas-krenn.com/en/wiki/Linux_I/O_Performance_Tests_using_dd) (use to write data to the drive and record the time it takes).  Example:  **dd if=/dev/zero of=testWriteSpeed.txt bs=1G count=1** | [Screenshot](https://drive.google.com/file/d/1UxMxgb307WiIyspzbHLiEfDrT3HC9gIF/view?usp=sharing) |
| Cooperative play cluster latency | <30 milliseconds | Give the time to reach the cluster. For example:  **rtt min/avg/max/mdev = 0.011/0.027/0.049/0.014 ms** | Ping localhost or google.com from one of the instances | [Screenshot](https://drive.google.com/file/d/1HzZiDpFDuusi-k3yBdtH5Tvmw8aqHCZB/view?usp=sharing) |
| Cooperative play latency between gateway/matching and core | <30 milliseconds | Give the time to go from gateway/matching and core. For example:  **rtt min/avg/max/mdev = 0.011/0.027/0.049/0.014 ms** | Ping localhost or google.com from one of the instances | [Screenshot](https://drive.google.com/file/d/1HzZiDpFDuusi-k3yBdtH5Tvmw8aqHCZB/view?usp=sharing) |
| Cooperative play latency between gateway/matching and environment | <30 milliseconds | Give the time to go from gateway/matching and environment. For example:  **rtt min/avg/max/mdev = 0.011/0.027/0.049/0.014 ms** | Ping localhost or google.com from one of the instances | [Screenshot](https://drive.google.com/file/d/1HzZiDpFDuusi-k3yBdtH5Tvmw8aqHCZB/view?usp=sharing) |
| Pull time from the cooperative play instances and initialization time | 15–30 minutes for each cluster | Provide time to start cluster (same as row 5). For example: | Kitchen create, kitchen converge, | [Screenshot](https://drive.google.com/file/d/1E8OjqjfnIyUVLkXuE3tqdXVBKBP6xcPs/view?usp=sharing)1  [screenshot2](https://drive.google.com/file/d/1x42mRhbozzCEbthhoLt7eFPfMuJ8MGU0/view?usp=sharing) |

## How to get the Data

As shown in the [learn.chef.io tutorials](https://learn.chef.io/modules/local-development/rhel/virtualbox/apply-a-cookbook#/), you can use either [**kitchen exec -c ‘COMMAND’**](https://docs.chef.io/ctl_kitchen.html#kitchen-exec) or you can ssh to the instance with [**vagrant ssh <instanceid>**](https://www.vagrantup.com/docs/cli/ssh.html) and run the command on the instance to get the output for each command provided in the diagnostic report example above:

|  |
| --- |
|  |

For example, you can use the instance name from the [**kitchen.yaml**](https://docs.chef.io/config_yml_kitchen.html) file **or** from [**kitchen list**](https://docs.chef.io/ctl_kitchen.html) to identify the instance you want to run top on:

|  |
| --- |
|  |
|  |

Here is another way:

run [**vagrant global-status**](https://www.vagrantup.com/docs/cli/global-status.html) to get the vagrant ids of each instance in your environment:

|  |
| --- |
|  |

Lets say I want to get the output of [**top**](http://man7.org/linux/man-pages/man1/top.1.html) (which will give me the system load) from my **cgsp-2-core** server/instance. I would run [**vagrant ssh 1bcc1d1**](https://www.vagrantup.com/docs/cli/ssh.html) to ssh to it:

|  |
| --- |
|  |

Then I can run [**top**](http://man7.org/linux/man-pages/man1/top.1.html) and take a screenshot of the output. The load averages are **0.00, 0.01, 0.05, so I would record this in the data column for row 3, column 3** :

|  |
| --- |
|  |

You can refer to [tops manpage](http://man7.org/linux/man-pages/man1/top.1.html), [this resource](https://www.howtogeek.com/194642/understanding-the-load-average-on-linux-and-other-unix-like-systems/), or [use google](https://www.google.com/search?q=top+load+average) for more info on how to read the output of top:

|  |  |  |
| --- | --- | --- |
|  |  |  |

# E. Web Sources

Provide the web sources you modified to create both the automation script and the scripts used to gather the performance measurements. You are free to use any scripts from stack overflow, github, or other sources, just make sure you cite each one here (as a link) if you used part or all of it for your solution. If you used system commands like ping, top, etc, just use a link to a reference that describes how to use the command. You should have **at least 3**. For example:

|  |
| --- |
| * Docker Kitchen Driver​ - Used to spawn docker clusters within test kitchen.   + https://github.com/test-kitchen/kitchen-docker * Learn.chef.io ​- Getting Started with Test Kitchen - Used as guidance when configuring test kitchen.   + https://learn.chef.io/modules/local-development/ubuntu#/infrastructure-automation * Apache 2 Cookbook​ - Used for configuring automation of web front end and back end.   + https://github.com/sous-chefs/apache2 * SQL Lite Cookbook ​- Used for configuring automation of database cluster.   + https://github.com/ualaska-it/sqlite\_install * Chef Supermarket​ - Chef Cookbooks - Used to find cookbooks used for installation.   + https://supermarket.chef.io/cookbooks-directory * Docker Documentation​ - Used as guidance while configuring docker containers.   + https://docs.docker.com/get-started/ * Linux Training Academy​ - Command Line Cheat Sheet - Used as a reference for locating Linux command-line tools/scripts for testing performance.   + https://www.linuxtrainingacademy.com/linux-commands-cheat-sheet/ * Performance metrics were supplied using the notated Linux applications.   + Nmon - ​<http://nmon.sourceforge.net/pmwiki.php>   + Iostat - ​<https://linux.die.net/man/1/iostat>   + Ping - ​https://linux.die.net/man/8/ping * https://kitchen.ci/docs/getting-started/introduction/ * ○ Syntax web source used for kitchen.yml   + https://docs.chef.io/config\_yml\_kitchen.html * ● Documentation for ZeroMQ was obtained from:​ ​http://zguide.zeromq.org/page:all |

# F. Sources

Please include in-text citations for sources that are properly quoted, paraphrased, or summarized and a reference list that accurately identifies the author, date, title, and source location as available. You can use APA, MLA, or Chicago citation standards. You can use [bibme.org](https://www.bibme.org/), [citation machine](https://www.citationmachine.net/), or [easybib.com](https://www.easybib.com/) to format the citations for you.

# G. Professional Communication

Demonstrate professional communication in the content and presentation of your submission. Just dot your i’s and cross your t’s here and make sure it reads like something you would present to your employer.

# How to submit your solution

Your submission should use and include all of the files provided in the zip archive you downloaded from the supporting documents section. In addition, it should include all of the scripts used to complete the diagnostic report document and a copy of your cookbook including the kitchen.yml file.

# Errors and fixes

## Error parsing kitchen.yml

|  |
| --- |
|  |

Make sure you watch your spacing in your kitchen.yml. You should two spaces next to – name.

Reference: <https://stackoverflow.com/questions/33149008/getting-error-for-kitchen-create-in-chef>

## Could not load the chef provisioner

|  |
| --- |
| Log file: |

You have typos in the kitchen.yml (<https://stackoverflow.com/questions/33652688/could-not-load-the-chef-solo-provisioner-from-the-load-path>).

## Unable to ping from VM1 to VM2

You may have to **add** a new network adapter in virtualbox for your VMs. The default is usually **NAT**, which allows VM -> Host, VM -> Net/LAN connectivity. This mode can allow for VM <- host and VM <- Net/LAN connectivity, but port forwarding must be enabled. NAT cannot do VM1 <->VM2 connectivity, but you should keep one interface with NAT as vagrant uses this to connect to and manage your instances. You must **add Bridged** or **Host only** mode, which allows for more advanced networking. Both modes can be enabled by selecting the respective setting below in the Virtualbox gui for **each** VM:

|  |
| --- |
|  |

You can refer to [chapter 6](https://www.virtualbox.org/manual/ch06.html) of the virtualbox manual for more info about each of the network modes in virtualbox.

|  |
| --- |
|  |

You can also do this by using the **–[public\_network] or –[private\_network]** parameter in the **kitchen.yml** file so that the second interface is added for each VM when **kitchen create** is ran. Please refer to the kitchen.yml examples in [how to setup your environment](#_rv0e6txfpbfj) or [C. Automation Script](#_p03hei523bag) for examples.

Use ifconfig and ipconfig on linux and windows respectively to verify network configuration:

|  |  |
| --- | --- |
|  |  |

You can verify connectivity between VMs with ping:

|  |
| --- |
|  |

## **Stderr: VBoxManage.exe: error: Not in a hypervisor partition (HVP=0) (VERR\_NEM\_NOT\_AVAILABLE).**

|  |
| --- |
| -----> Starting Test Kitchen (v2.9.0)  -----> Creating <default-centos-7>...         Bringing machine 'default' up with 'virtualbox' provider...         ==> default: Checking if box 'bento/centos-7' version '202012.21.0' is up to date...         ==> default: Setting the name of the VM: kitchen-learn\_chef\_httpd-default-centos-7-16cb2d38-d462-4768-9f3a-00a23b565efa         ==> default: Clearing any previously set forwarded ports...         ==> default: Clearing any previously set network interfaces...         ==> default: Preparing network interfaces based on configuration...             default: Adapter 1: nat         ==> default: Forwarding ports...             default: 22 (guest) => 2222 (host) (adapter 1)         ==> default: Running 'pre-boot' VM customizations...         ==> default: Booting VM...         There was an error while executing `VBoxManage`, a CLI used by Vagrant         for controlling VirtualBox. The command and stderr is shown below.         Command: ["startvm", "e152d6a6-a3e6-4534-aa26-c591e5fc839c", "--type", "headless"]         Stderr: VBoxManage.exe: error: Not in a hypervisor partition (HVP=0) (VERR\_NEM\_NOT\_AVAILABLE).         VBoxManage.exe: error: AMD-V is disabled in the BIOS (or by the host OS) (VERR\_SVM\_DISABLED)         VBoxManage.exe: error: Details: code E\_FAIL (0x80004005), component ConsoleWrap, interface IConsole  >>>>>> ------Exception-------  >>>>>> Class: Kitchen::ActionFailed  >>>>>> Message: 1 actions failed.  >>>>>>     Failed to complete #create action: [Expected process to exit with [0], but received '1'  ---- Begin output of vagrant up --no-provision --provider virtualbox ----  STDOUT: Bringing machine 'default' up with 'virtualbox' provider...  ==> default: Checking if box 'bento/centos-7' version '202012.21.0' is up to date...  ==> default: Setting the name of the VM: kitchen-learn\_chef\_httpd-default-centos-7-16cb2d38-d462-4768-9f3a-00a23b565efa  ==> default: Clearing any previously set forwarded ports...  ==> default: Clearing any previously set network interfaces...  ==> default: Preparing network interfaces based on configuration...      default: Adapter 1: nat  ==> default: Forwarding ports...      default: 22 (guest) => 2222 (host) (adapter 1)  ==> default: Running 'pre-boot' VM customizations...  ==> default: Booting VM...  STDERR: There was an error while executing `VBoxManage`, a CLI used by Vagrant  for controlling VirtualBox. The command and stderr is shown below.  Command: ["startvm", "e152d6a6-a3e6-4534-aa26-c591e5fc839c", "--type", "headless"]  Stderr: VBoxManage.exe: error: Not in a hypervisor partition (HVP=0) (VERR\_NEM\_NOT\_AVAILABLE).  VBoxManage.exe: error: AMD-V is disabled in the BIOS (or by the host OS) (VERR\_SVM\_DISABLED)  VBoxManage.exe: error: Details: code E\_FAIL (0x80004005), component ConsoleWrap, interface IConsole  ---- End output of vagrant up --no-provision --provider virtualbox ----  Ran vagrant up --no-provision --provider virtualbox returned 1] on default-centos-7  >>>>>> ----------------------  >>>>>> Please see .kitchen/logs/kitchen.log for more details  >>>>>> Also try running `kitchen diagnose --all` for configuration |

If you get the above error, make sure hyper-V is disabled on your computer.  You may also need to go to bios settings and make sure cpu virtualization is enabled as shown in the following videos:

[For intel cpus and most bios](https://www.youtube.com/watch?v=18fU8FHgG80&ab_channel=SharingKnowledgeVideo)

[For MSI bios (enable SVM mode under OC settings / CPU)](https://www.youtube.com/watch?v=OtTkbfscU3g&ab_channel=furulevi)

## Packer errors using windows server 2012

If you are using packer and windows server 2012 in place of centos, you may need to run these on your new 2012 Server image, powershell admin promp:

* Set-Item WSMan:\localhost\Service\AllowUnencrypted -Value True
* Set-Item WSMan:\localhost\Service\Auth\Basic -Value True

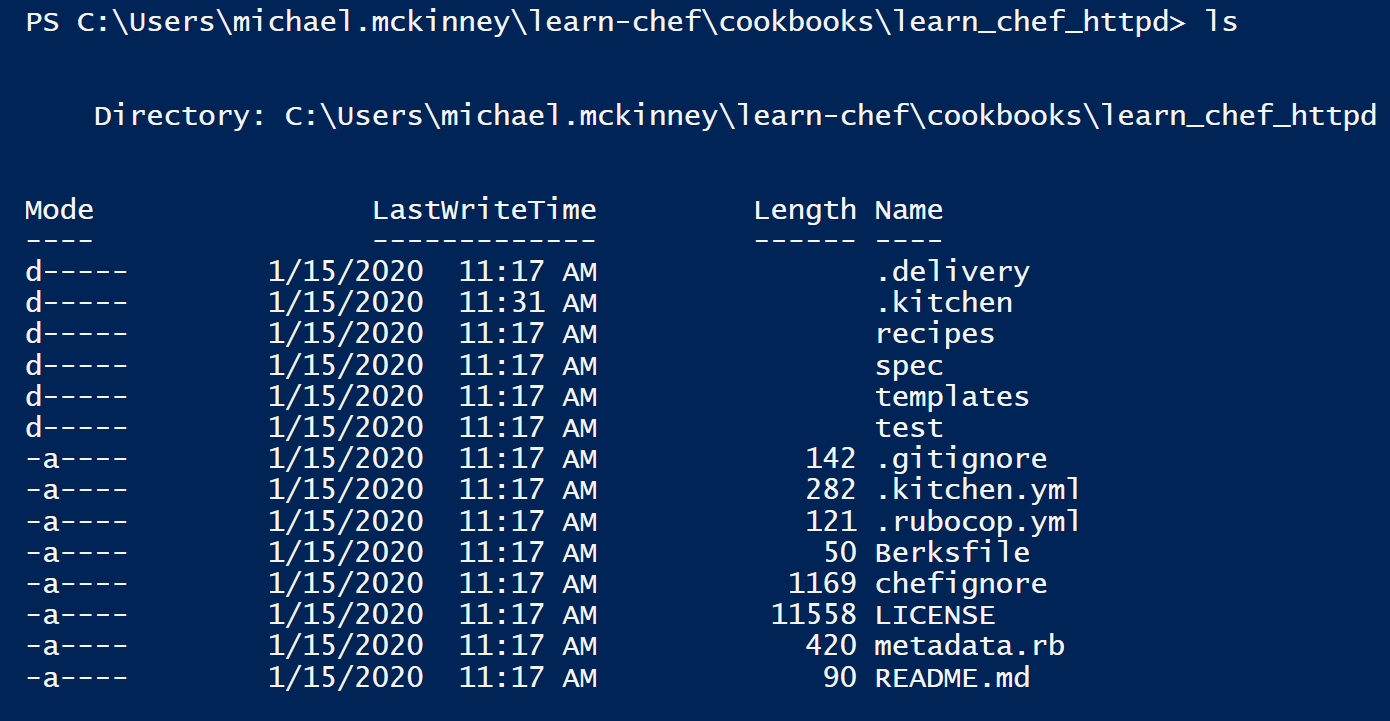
If you are getting endless "Waiting for WinRM" on your packer builds, try these two commands from powershell admin prompt on your VirtualBox Image:

* Set-Item WSMan:\localhost\Service\AllowUnencrypted -Value True
* Set-Item WSMan:\localhost\Service\Auth\Basic -Value True

## Chef Workstation cannot execute without accepting the license

* Accept the chef license by following [this article](https://docs.chef.io/chef_license_accept.html)

# How to get help

* You can schedule an appointment with your instructor using the “Schedule appointment” link below or on the course landing page, or send us an email with any questions, concerns, or assistance that is needed so we can get this course finished!
* When sending an email, please include a **full screenshot** showing the error or issue you are running into, and attach a **zip** **archive** of your [cookbook](https://docs.chef.io/cookbooks.html). Your cookbook is the directory with your **.kitchen.yml** file and may look like the learn\_chef\_httpd cookbook in the example below.  
  
* You can still send an email to your instructor when they are out of office and they will respond as soon as possible.
* You can also send an email to [itcloud@wgu.edu](mailto:itcloud@wgu.edu) with the subject **D085 - <subject>** and cc your instructor on it to gain assistance if they are out of the office as well.

Thank you for your time, and our team looks forward to working with you!