Red Hat
Training and
Certification

Write a Student Guide

Table of Contents

Introduction

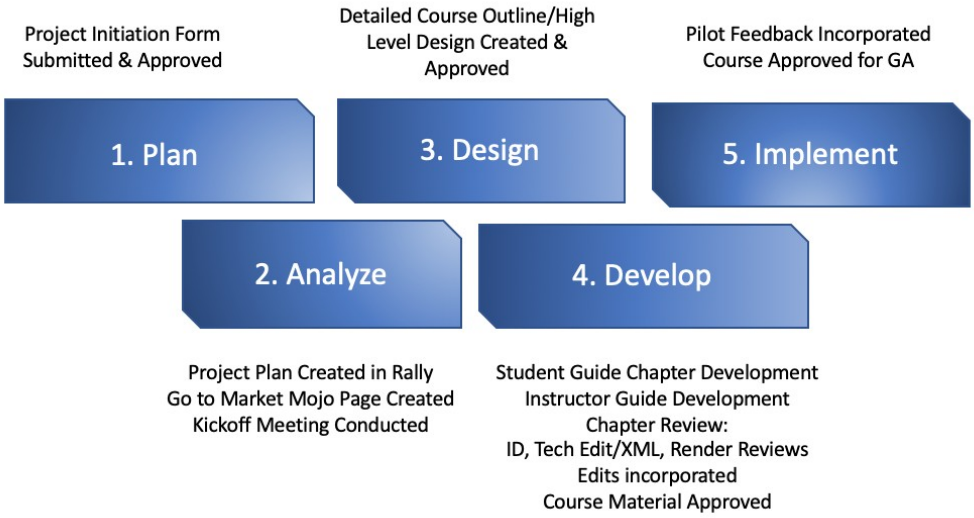
This resource contains guidelines for creating and designing GLS training courses. Included in this document are details about the phases of the course development process and the documentation that is required during each phase. Instructions, tips, examples, and links for templates are also included.

Developing effective, high-quality courses is often challenging when working with multiple content developers. Creating a high-quality course begins with developing and maintaining consistent course design, writing, and adherence to tagging standards. Consistent standards help students develop a feeling of familiarity when taking Red Hat Training courses.

Read these guidelines from beginning to end if you are a new content developer at Red Hat. If you are more experienced, scan the table of contents to find the information you need. Refer back to these guidelines as you build your courses.

This document is a working document and will be updated regularly as needed.

If you have additional questions about these guidelines, email Dave here: dsacco@redhat.com.



Course Development Process

The GLS course development process is divided into five major phases. It has been adapted from the ADDIE instructional systems design model for the GLS course development process. The individual tasks for the course development phase appear in Rally, the GLS Curriculum Teams project management tool.

GLS Course Development Review Process

During each phase of your development, your work product will be reviewed. There are three major types of reviews performed by Content Services. Detailed information about each type of Content Services review is found in the Review the Content Services Editing Process document.

1. Instructional Design review – Fit within curriculum, audience identified, Objectives/ Chapter/Section/Titles are clear and measurable, timing verified.
2. Technical Editing/XML tagging review - Basic proofreading—spelling, punctuation, style conventions, grammar, syntax, sentence construction consistent, active voice used, tone, flow; XML tagging review—compliance to tagging standards.
3. Chapter Render reviews happen after ID/Technical XML Tagging reviews occur and focus on print issues, text overflow, rendering, and images. Once content is updated and approved, the author builds slides and submits them for review by a member of Content Services. Finally, the translation process focuses on rendering issues and missing items after vendor translations are sent back to the Content Services Project Manager.

Content Flow from Author to Content Services

You will create narratives, labs, and exercises as Student Guide Chapters > Submit individual chapter PDF and Repository links in Rally task (**ID review task**) > ID reviews SG content in XML and adds comments in-line > ID Sets (**Technical editing and XML tagging review)** task to Ready > Technical Editor reviews XML and incorporates comments in-line > Technical Editor sets **(Apply technical and ID editing changes)** task to Ready > Author applies changes, renders PDF > Sets **(PDF render review)** task to Ready.

Storing Project Forms

As part of the course development process, several project forms are created. You or your team lead will need to create a project storage location within the GLS Curriculum Team Mojo site if one has not been created already. Be sure to check the site and consult with your scrum master before creating a project storage location.

Creating a New Project Storage Location

1. Go to the GLS Curriculum Team Mojo page: https://mojo.redhat.com/groups/curriculum-team/overview
2. Click the Projects tab.
3. At the upper right of the screen, click Actions > Project
    a. Name the project, e.g., RH236.

b. Add a project description, e.g., Storage Administration.
c. Set an approximate start/end date for the project.
d. Make sure the team member who will be creating the curriculum development forms is set as project owner.

4. Click Preview.
5. Click Create Project (if you get a warning message "Null" ignore it).

Adding a Form to Your Project Storage Location

1. Go to the GLS Curriculum Team Mojo page: https://mojo.redhat.com/groups/curriculum-team/overview.
2. Choose your project from the list on the left side of the page.
3. Click Uploaded Files.
4. Browse to the form file.
5. Click Publish.

The Course Development Project Initiation Form

The Course Development Project form provides management teams with course and project-specific planning information, including business, technical, and subscription justifications for creating or updating a course. This information will support a decision of whether or not to pursue a development project, and results in a go or no-go decision to move forward to the next phase in development. The Curriculum Architect (CA) typically completes this form, with reviews by management. This must be completed and signed off on prior to course development.

Choose **Example: Course Development Project Initiation Form** from this link. Choose **Template: Course Development Project Initiation Form** from this link.

The Objectives Analysis

Your Curriculum Architect, and possibly the Team Lead, may meet with peers, support engineers, consultants, partners, and other teams or individuals who will attend your course. From those meetings and research, they will obtain the critical skills/knowledge of the target audience needed to create the Objectives Analysis portion of the Detailed Course Outline (DCO) of High Level Design (HLD) form. The Objectives Analysis portion lists each chapter number, name, goal, section titles and objectives. The Curriculum Architect typically completes these entries, with reviews by management, Instructional Designer, and Technical Editor. This must be completed and signed off on prior to course development.

Items in the Objectives Analysis

Physical Course Structure

Red Hat Training courses are comprised of chapters. Chapters are broken into sections, and sections are broken into paragraphs of text, screen captures of code, graphics, lists, tables, etc. Each chapter begins with an introductory table page to outline the chapter. Each chapter ends with a summary page that restates the key points of the chapter.

Writing Chapter Titles

Write chapter titles when filling out the Objectives Analysis portion of the DCO. Choose descriptive language that matches the chapter content. Knowledge-based and Skills-based chapter titles begin with a gerund (an -ing verb).

Example of a knowledge-based chapter title: "Networking for Port Security"
Example of a skills-based chapter title: "Controlling Services and Daemons"

Learning Objective Gerund Google Sheet. Use this sheet to aid in developing titles.

Writing Goals and Objectives

Writing goals and objectives is the first step in designing an educational experience that will be meaningful and motivating for the students. Goals and objectives are essential to effective instruction. Use the list of tasks and skills that you determined from the informational meetings and research you've performed to write goals and objectives. Keep the chapter or course goal in mind as you identify the content you will include in the course or chapter. Write goals to inform the students of exactly what they will learn after completing the chapter or course.

Goal/Objective and Placement	Example
Course Goal GTM Mojo page	This course prepares the student to install Red Hat JBoss Data Grid, configure caches, and develop Java applications that utilize a data grid.
Course Objectives Student guide—About this Course page Instructor guide—Introduction page	• Gain sufficient skill to perform core system administration tasks on Red Hat Enterprise Linux. • Build foundational skills needed by an RHCSA-certified Red Hat Enterprise Linux system administrator
Chapter Objectives/Goals Course Mojo page—Course outline Student guide—Introductory table of each chapter	Set Linux file system permissions and describe the security effects of permission settings
Section Objective(s)	After completing this section, you should be

Goal/Objective and Placement	Example
Student guide Instructor guide	able to: • Install Qpid. • Perform a basic configuration. • Install Qpid tools.

Writing Course Goals

Write the course goal to state what you want the student to know how to do as a result of taking the course. The course goal is a broad, course-wide statement. Write the course goal when filling in details for the course Mojo page.

Writing Course Objectives

Write course objectives for the student guide Introduction page. Write course objectives that clearly convey what the student will learn as a result of taking the whole course. State the general purpose or benefits of taking the whole course. Include an overall reference to how well a student must perform tasks or skills. Use the course description to guide you when writing course objectives.

Writing Chapter Objectives/Goals

Write the chapter objective/goal when completing the Objectives Analysis form. Chapter objectives/goals are listed on the course wiki page under the Course Outline area and in the goal column of the introductory table. Write the chapter objective/goal after you list the section objectives (tasks) on the Objectives Analysis form, and then combine the section objectives into a sentence that focuses on the overarching, higher-level topic of the chapter. You do not need to list every task covered in a chapter when writing the chapter objective/goal. Chapter objectives/goals must start with an action verb.

Writing Section Objectives

Create section objectives when completing the Objectives Analysis form. Insert the section title and fill out the section objectives when creating each section. Use the DCO as your guide. Section objectives describe skills the student will be able to perform after completing the section.

Section objectives start with a verb. Section objectives include:

- **Introductory objective text**: "After completing this section, you should be able to… xxx." All section objectives start with this text.
- **Action:** List the skills, knowledge, and actions students will perform in the section. Begin all section objectives with an action verb that is specific to the skill they will perform. If you have one section objective, title the section Objective and use a run-on sentence; otherwise, use a bulleted list (see following example).

Note: For sections with one (1) objective, be sure to write it as one sentence with no bullet: "After completing this section, you should be able to install Ansible Tower."

Writing with Verbs

Choose clear, observable, and **measurable** verbs in your objectives and titles to communicate to the student precisely what behavior the student should be able to demonstrate after completing the chapter or section. Ask this simple question when choosing verbs: "Can I measure or quantify "understanding" or "discovering"? The answer is no, BTW. Use the following examples to guide you:

Effective verbs for knowledge-based sections

These verbs are specific to learning information (concepts, facts, definitions) and work well in technical content:

Explain, describe, define, list, identify…

Effective skill-based verbs for task sections

These verbs describe a task or activity that students will do. They clearly state what action you want students to do:

Manage, create, install, build, configure, test, connect, modify, deploy, troubleshoot, design, validate, monitor, import, export, copy, move, add, remove, edit, delete, migrate, archive, download, extend, boot, reboot…

For a more complete list of knowledge-based and Skills-based verbs, visit this document. We recommend you bookmark and use it each time you write a chapter.

Weak verbs

Since objectives should be measurable, avoid the following words because they are not specific enough, are not measurable, and are not observable:

Understand, comprehend, appreciate, discover, conceive, learn, use, utilize, review, examine, provide, work with…

The Detailed Course Outline and/or High Level Design Documents

An effective Detailed Course Outline (DCO) or High Level Design (HLD) will provide you and other content writers with a valuable reference to point to when writing your course. Curriculum Architects build the Objectives Analysis into the DCO and add details to the columns for section titles, section timing, section tasks, methodology/engagement methods, and activities when it's time to build out the DCO. Reference materials and Architect notes are added for you to use when writing the course. The DCO is completed prior to the actual development of course materials by the Curriculum Architect and is part of the official record of the course. The Curriculum Architect grants writers' access to the DCO as the formal outline and activity map used for development of each course. The Content Services team uses the DCO to review your course for proper objective, learning activity, and content alignment during development.

Note: Obtain approval of the DCO from your Curriculum Architect, Scrum Master and the Instructional Designer. The tech editor will also provide feedback on the technical accuracy of all chapter and section titles as part of the ID review approval process.

The High Level Design may be substituted for the DCO in some cases but instead of being formatted as a Google sheet, it's a continuous Google doc. The decision to use one vs. the other is driven by business need and Curriculum Architect preference, but both should include similar elements. The more thorough document is the DCO because it allows for exercises and section numbering. For this reason, we go in more depth on DCO's below:

Tips for Completing the DCO

This section is meant for Curriculum Architects, but is important to your understanding of the Design process and how it influences course development. CA's complete the DCO Sheet by:

- Adding section titles.
- Choosing the instructional methods for each section.
- Choosing activities for each section.
- Listing existing content and reference material.
- Identifying opportunities for Expert Seminars, per section. We are targeting 2 per chapter.

Choose: **"Example: Detailed Course Outline"** from this link

Writing Section Titles

Write section titles that are descriptive and match the section content. Knowledge-based section titles do not start with a verb. Write task-based section titles beginning with a gerund (an -ing verb.) Create your section titles when completing the DCO.

Example of a section title: "Entering Commands Using CLI."

Choosing Instructional Methods

When completing the DCO, choose the appropriate instructional methods:

- **Engagement methods**—Used when you are instructing or presenting information to the students:
  - Lecture
  - Activities
- **Activities** — Used when you want the students to practice, apply, or respond to instruction:
  - Demonstration (instructor-led, view-only demo of the most important aspects of a process or procedure).
  - Guided exercise (self-paced, in class).
  - Guided exercise (formerly workshop – performed along with the instructor).
  - End of Chapter Lab (minimal steps, designed to assess a learner's knowledge of key skills or concepts from each chapter; solutions follow).
  - Comprehensive review (culmination of all skills/knowledge gained in a course, in the form of a series of labs; solutions follow).

- **Knowledge check methods**—Used when you want the students to check their understanding of what has been taught in a knowledge-based chapter. Any chapter that includes a critical skill must have an end of chapter lab. For knowledge-only chapters, the types of checks include:
    - Multiple-choice quiz
    - Sequencing quiz
    - Matching quiz

Engagement Methods

Writing Lectures

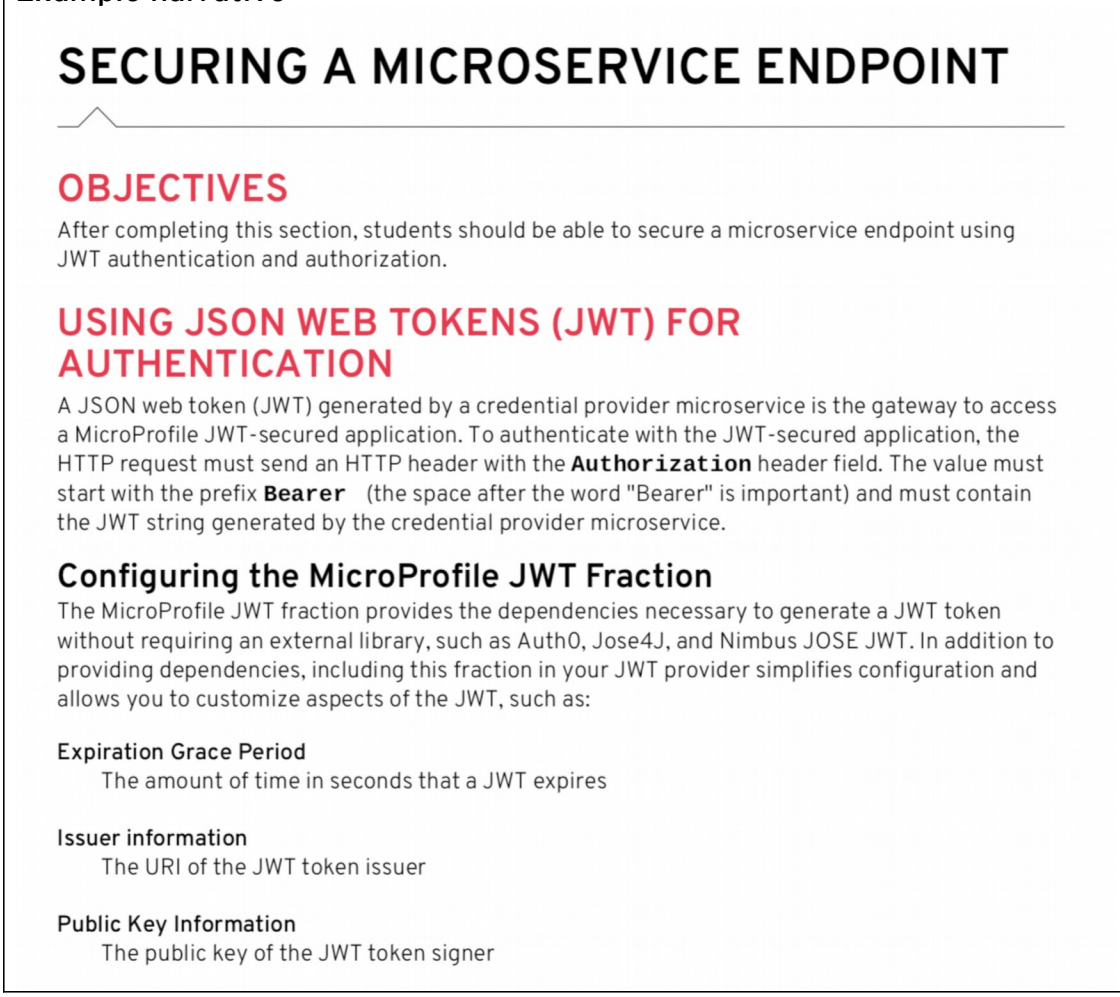
A lecture is the most common instructional method used by GLS Curriculum teams.

Note: Standard practice is to write Activity Methods (Guided Exercises and End-of-chapter labs) *prior* to the narrative of the course. See Activity Methods for more information.

Write a lecture when you want to deliver content to the student. Segment your lecture content by including:

- Headings
- Graphics
- Screen output
- Numbered lists
- Bulleted lists

Example narrative



SECURING A MICROSERVICE ENDPOINT

OBJECTIVES

After completing this section, students should be able to secure a microservice endpoint using JWT authentication and authorization.

USING JSON WEB TOKENS (JWT) FOR AUTHENTICATION

A JSON web token (JWT) generated by a credential provider microservice is the gateway to access a MicroProfile JWT-secured application. To authenticate with the JWT-secured application, the HTTP request must send an HTTP header with the **`Authorization`** header field. The value must start with the prefix **`Bearer`** (the space after the word "Bearer" is important) and must contain the JWT string generated by the credential provider microservice.

Configuring the MicroProfile JWT Fraction

The MicroProfile JWT fraction provides the dependencies necessary to generate a JWT token without requiring an external library, such as Auth0, Jose4J, and Nimbus JOSE JWT. In addition to providing dependencies, including this fraction in your JWT provider simplifies configuration and allows you to customize aspects of the JWT, such as:

Expiration Grace Period
The amount of time in seconds that a JWT expires

Issuer information
The URI of the JWT token issuer

Public Key Information
The public key of the JWT token signer

Activity Methods

Writing Guided Exercises

Write a guided exercise to offer your students an opportunity to practice tasks on their own that have been taught in a previous section. The first actual development you will create for a course is a guided exercise for each skill-based (task) section. Guided exercises include both steps and detailed instructions on how to execute those steps. Students perform the exercise steps during class time. A guided exercise includes:

- **Guided Exercise title**: For example, “Building SSH Key-based Authentication”
  Note: The title should begin with a gerund.
- **Guided Exercise objective**: “In this exercise, you will ... xxx.”
- **Outcome statement:** “You should be able to … xxx.”
  Note: For more detail about outcome statements, see “Writing Outcome Statements” later in this document.

- **Prerequisite information or actions**: "*Before you begin…*"
- **Include this final sentence for all guided exercises:** "This concludes the guided exercise."

Guided Exercise tip: When writing steps for your exercises and labs, be aware of the amount of context provided to your learner. Often times, not enough explanation is provided within a step. Here are some examples and suggestions for writing more effective exercise and lab steps:

Not enough information in a step:

1. Launch an instance.
[root@servera ~]# openstack server create --flavor m1.small --key-name demokeypair --image demoimage --nic net-id=demonet demovm

The problem is that the student is not given enough information in the step to successfully perform the command. Writing steps in this way encourages learners to blindly type in the command without thinking; something we want to avoid.

A better way to write this step would be:

1. Launch an instance named demovm using the m1.small flavor. Choose the demokeypair key pair, the demoimage image, and the demonet network.

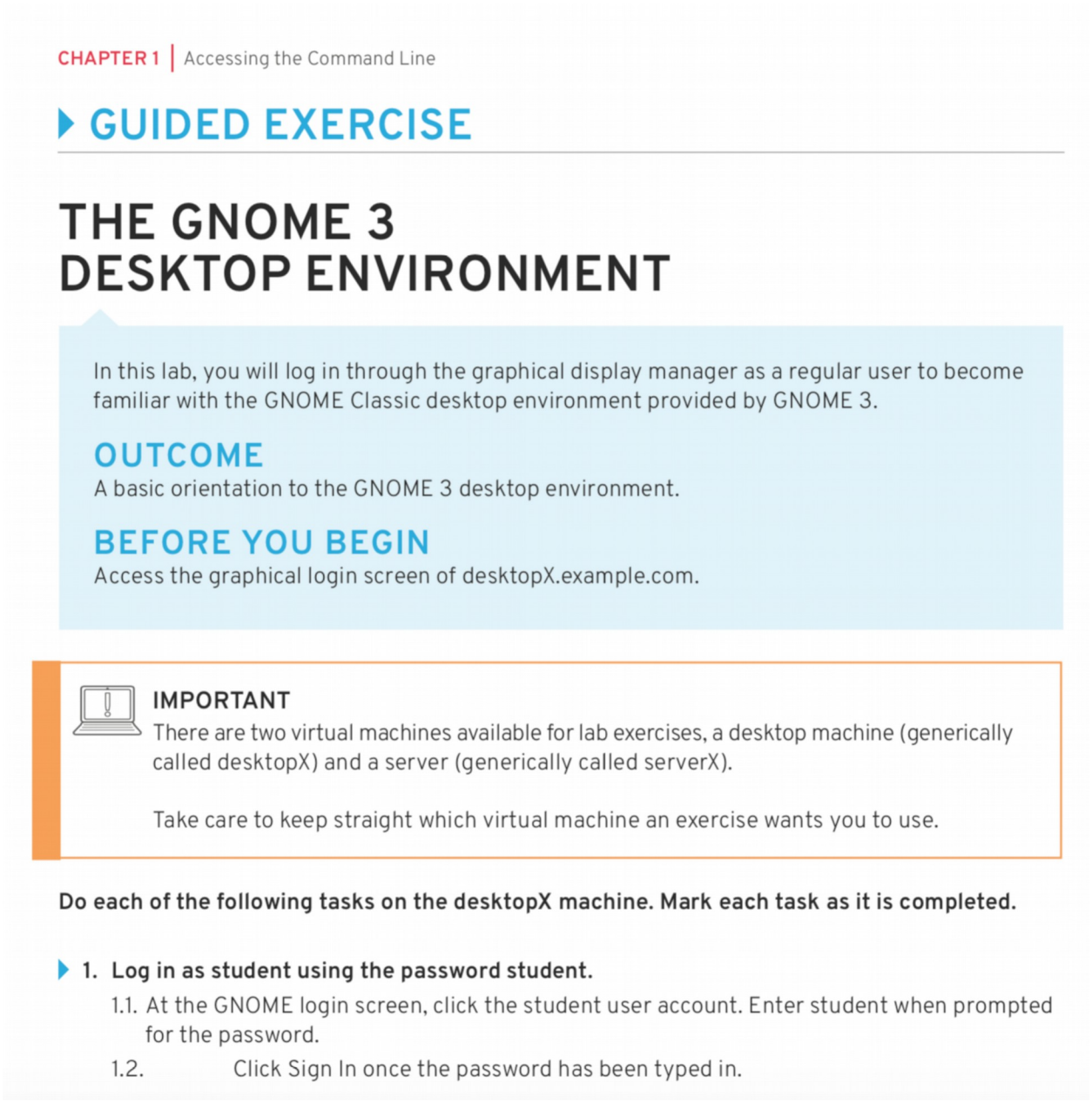
A second example I see in steps with sub steps is that the first para should explain all of the sub-steps:

1. Launch an instance named testvm.
    1.1. Find the network…
    1.2. Launch the testvm…
    1.3. Verify the instance…

The first step should provide enough information for someone to perform the sub-steps. In this case, a better first step would be:

1. Find the network name in order to launch the testvm instance. Once the instance is booted, verify the instance.

Example Guided Exercise



CHAPTER 1 | Accessing the Command Line

GUIDED EXERCISE

THE GNOME 3 DESKTOP ENVIRONMENT

In this lab, you will log in through the graphical display manager as a regular user to become familiar with the GNOME Classic desktop environment provided by GNOME 3.

OUTCOME

A basic orientation to the GNOME 3 desktop environment.

BEFORE YOU BEGIN

Access the graphical login screen of desktopX.example.com.

IMPORTANT

There are two virtual machines available for lab exercises, a desktop machine (generically called desktopX) and a server (generically called serverX).

Take care to keep straight which virtual machine an exercise wants you to use.

Do each of the following tasks on the desktopX machine. Mark each task as it is completed.

1. **Log in as student using the password student.**
   1.1. At the GNOME login screen, click the student user account. Enter student when prompted for the password.
   1.2. Click Sign In once the password has been typed in.

This concludes the guided exercise.

Writing End-of-chapter Labs

Write a lab that provides a performance checklist at the end of the chapter to assess student comprehension. Give your students a list of tasks to perform without additional instructions. Include an instruction (or step) for each task taught in the sections. Whenever possible, build your end-of-chapter lab around an example, analogy, simulation, or case study that directly relates to the task being taught. This will help students retain knowledge and relate it to a job skill. An end-of-chapter lab includes:

- **Lab Title**: For example, “Testing Microservices.”

Note: The title of the end-of-chapter lab **must** match the *chapter* title and start with a gerund.)

- **Lab objective**: "In this lab, you will ... xxx."
- **Outcome statement**: "You should be able to … xxx."
- **Prerequisite information or actions***: "Before you begin…"*
- **Lab steps with minimal/no instructions:** Only include critical information for the learner to perform the lab. These should not be as verbose as Guided Exercise steps. The idea is to assess the learner's comprehension of key objectives, not to guide them through each portion of a step.
- **After the last step of every end-of-chapter lab, include this sentence:** "This concludes the lab."

Example end-of-chapter lab

LAB

TESTING MICROSERVICES

PERFORMANCE CHECKLIST

In this lab, you will implement an integration test and a unit test using mock frameworks for the microservice-speaker application.

OUTCOMES

You should be able to implement an integration test and a unit test using Arquillian, Wiremock, Rest Assured, and Hamcrest.

BEFORE YOU BEGIN

If you have not already done so, use **`git clone`** to download the microprofile-conference repository to the **`workstation`** machine.

```
[student@workstation ~]$ git clone \
http://services.lab.example.com/microprofile-conference
Cloning into 'microprofile-conference'...
...output omitted...
Resolving deltas: 100% (2803/2803), done.
```

Then run the lab setup to begin the exercise.

```
[student@workstation ~]$ lab test-review setup
```

1. To begin the exercise, change to the **`lab-test-review`** branch of the application code.

```
[student@workstation ~]$ cd microprofile-conference
        [student@workstation microprofile-conference]$ git checkout lab-test-
review
Switched to branch 'lab-test-review'
```

1.1. Switch to the branch using the **`git checkout`** command.

```
[student@workstation ~]$ cd microprofile-conference
[student@workstation microprofile-conference]$ git checkout lab-test-
review
... output omitted ...
Switched to a new branch 'lab-test-review'
```

This concludes the lab.

Writing a Comprehensive Review

Write an end-of-course comprehensive review to prove your student's ability to solve a real-world problem using the solutions taught and practiced in the course. An end-of-course comprehensive review is required at the end of any Red Hat course with a certification exam. All end-of-course comprehensive reviews should include:

- **Title Page:** "Comprehensive Review". Do not include the course title.
- **Goal Statement:** Course goal, pulled from the DCO or HLD.
- **Objectives:** "Review tasks from <course title>."
- **Sections:** "Comprehensive Review".
- **Lab:** Lab titles using includes will appear as a bullet list:
    - "Lab: Describing Microservice Architectures".
    - "Lab: Adding Users and Teams".
    - "Lab: Creating a Custom Dynamic Inventory".

For each lab, include:

- **Lab title:** Using includes, will be something like this: "Restoring Ansible Tower from Backup".
- **Outcome statement:** "You should be able to restore the Ansible Tower configuration from a backup."
- **Before you begin:** "In this exercise, you will restore the configuration of your Ansible Tower server using a backup archive".
- **Comprehensive review steps with minimal instructions:**
    - Note the chapters where knowledge/skills are found to individual sections of the Comprehensive Review. This allows learners to go back for a knowledge check.
    - Write steps with few clues or direction, other than what is absolutely necessary for the learner to perform the lab. The idea is to test the learner's comprehension of key objectives, not to provide them with guidance as in GE's. Consult with Dave (dsacco@redhat.com) if you have questions about this approach.
    - No more than 15 steps per section.

Example comprehensive review

COMPREHENSIVE REVIEW

OBJECTIVES

After completing this section, students should be able to review and refresh knowledge and skills learned in *Red Hat Application Development II: Implementing Microservice Architectures*.

REVIEWING RED HAT APPLICATION DEVELOPMENT II: IMPLEMENTING MICROSERVICE ARCHITECTURES

Before beginning the comprehensive review for this course, you should be comfortable with the topics covered in each chapter.

Do not hesitate to ask the instructor for extra guidance or clarification on these topics.

Chapter 1, *Describing Microservice Architectures*

Describe components and patterns of microservice-based application architectures.

- Define what a microservice is and the guiding principles for their creation.
- Describe the major patterns implemented in microservice architectures.

Chapter 2, *Deploying Microservice-based Applications*

Deploy portions of the course case study applications to an OpenShift cluster.

- Deploy a microservice from the MicroProfile Conference application to an OpenShift cluster.
- Deploy a microservice to OpenShift using the fabric8 Maven plug-in.

Chapter 3, *Implementing a Microservice with MicroProfile*

Describe the specifications in MicroProfile, implement a microservice with some of the specifications, and deploy it to an OpenShift cluster.

- Describe the specifications included in MicroProfile.
- Implement a microservice using the CDI, JAX-RS, and JSON-P specifications of MicroProfile.

Chapter 4, *Testing Microservices*

Implement unit and integration tests for microservices.

- Implement a microservice test case using Arquillian.
- Implement a microservice test using mock frameworks.

Chapter 5, *Injecting Configuration Data into a Microservice*

Inject configuration data from an external source into a microservice.

<Provide a solution after each section of Comprehensive Review lab>

Writing Effective Knowledge Checks

Writing Quizzes

Write a quiz when assessing the student's knowledge level. A quiz is required at the end of a knowledge-based section. Build questions to test against the section objectives.

Skills Assessments

Skills Assessments are used to determine entry-level skills and knowledge of potential customers for a given course. By completing the graded assessment, and depending on their score, the leaner will receive suggestions regarding which course(s) they are qualified to attend. Please note that these are only suggestions and not meant to be directive.

If the DCO indicates that questions will be used in a Skills Assessment, then you will be writing multiple-choice questions as directed by the DCO. Adhere to the quiz writing standards as outlined in this document.

Quizzes include at least one question for each section objective. Content developers may use the following quiz types:

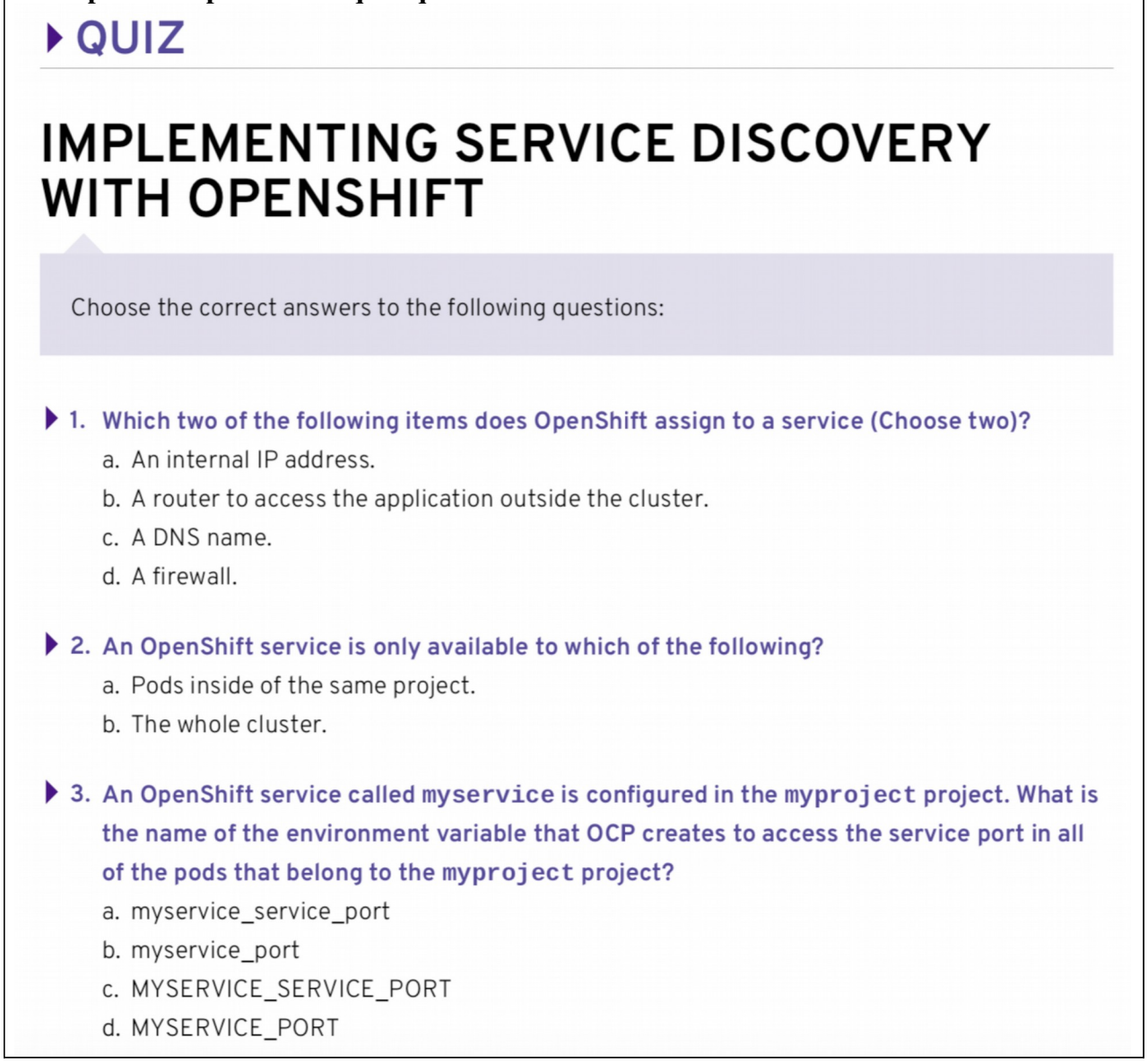
Quiz Type	Quiz Description
Multiple-choice	Use to measure the student's knowledge and skill. A multiple-choice quiz tests the student's ability to make a correct choice from a list of answers that all appear plausible. A well-written multiple-choice question can test the student's ability to apply knowledge and analyze information. **Note:** if you are writing a Skills Assessment, the only quiz type you can use is multiple choice.
Matching	Use when you want to assess the student's ability to identify parallel concepts, such as: • Terms and conditions • Causes and effects • Scenarios and responses
Sequencing	Use when teaching a process where order is important. A sequencing quiz tests the student's ability to answer sequentially.

All quizzes include:

- **Quiz title**
- **Instructions**
- **At least one multiple-choice, matching, or sequencing question per knowledge-based section objective**

The Content Services team has developed the document, Writing Quizzes.

Example multiple-choice quiz question



QUIZ

IMPLEMENTING SERVICE DISCOVERY WITH OPENSHIFT

Choose the correct answers to the following questions:

1. Which two of the following items does OpenShift assign to a service (Choose two)?
   a. An internal IP address.
   b. A router to access the application outside the cluster.
   c. A DNS name.
   d. A firewall.

2. An OpenShift service is only available to which of the following?
   a. Pods inside of the same project.
   b. The whole cluster.

3. An OpenShift service called `myservice` is configured in the `myproject` project. What is the name of the environment variable that OCP creates to access the service port in all of the pods that belong to the `myproject` project?
   a. myservice_service_port
   b. myservice_port
   c. MYSERVICE_SERVICE_PORT
   d. MYSERVICE_PORT

Example matching quiz

DESCRIBING MICROPROFILE AND ITS SPECIFICATIONS

Match the items below to their counterparts in the table.

CDI 1.2

Config 1.1

Fault Tolerance 1.0

Health Check 1.0

JAX-RS 2.0

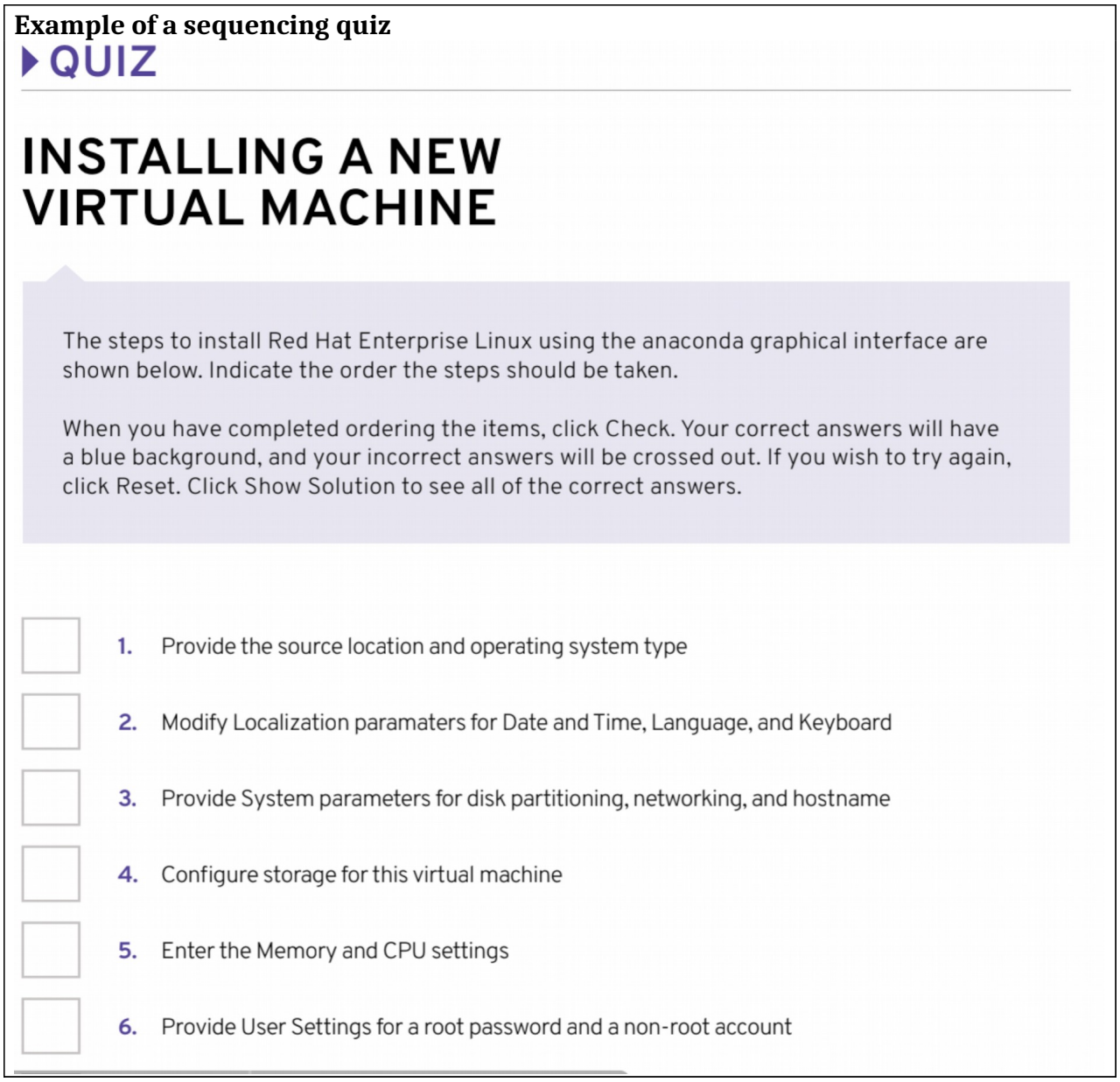
JSON-P 1.0

JWT Auth 1.0

Metrics 1.0

MICROPROFILE SPECIFICATION	DESCRIPTION
A specification that defines a set of complementary services that help improve the structure of application code including an enhanced lifecycle and interaction model over existing Java component types, including managed beans and Enterprise Java Beans.	
A specification that provides support in creating web services according to the Representational State Transfer (REST) architectural pattern using annotations to simplify the development and deployment of web service clients and endpoints.	

Example of a sequencing quiz



Developing Course Content

Writing a Student Guide

The bulk of what you write appears in the form of a student guide (SG). Keep in mind that you will also be editing and/or adding information and outputting different versions of the student guide, based on the three training modalities (described in greater detail as follows):

- Instructor-led Training (ILT)
- Virtual Training (VT)
- ROL (Red Hat Online Learning)

Writing Instructor Guides (IG) (ILT)

Create instructor guides to provide additional information to instructors about course content. IG's

The components of the Instructor Guide included in the skeleton are:
Overview: Goal statement for the chapter.
Schedule: ILT/VT schedule pulled in by includes in the XML, displaying the section titles, presentation & engagement methods, and time allotted to each.
Total Time: Pull this from the DCO or HLD.
Chapter Objectives: Pulled in by includes in the XML.
Key Takeaways: Lists the Summary bullets as an advanced organizer for instructors; a way to understand the key learning points from the chapter.
Instructor Tips and Suggestions: The section titles are pulled in by includes in the XML but the actual tips and sugReceiving objects: 15% (1070/6818), 3.50 MiB | 59.00 KiB/s gestions come from the author. Include items that would help the instructor set the students up for success. Note specific helpful hints or gotcha's to think about with exercises or narrative.

Example of an Instructor Guide:

Schedule

ILT/VT Schedule

Section	Title	Presentation & Engagement Methods	Time (minutes)
Introduction			3
1	Writing YAML inventory files	P: Lecture	15
		A: Guided Exercise	20
2	Managing Inventory Variables	P: Lecture	15
		A: Guided Exercise	15
	Managing Inventories	L: Review Lab	30
Conclusion			2

Total Time: 100 minutes

Chapter Objectives

- Write static inventory files in YAML format instead of the older INI-like format.
- Structure host and group variables using multiple files per host or group, and use special variables to override the host, port, or remote user Ansible uses for a specific host.

Key Takeaways

- How to write static YAML inventory files.
- How to convert a INI inventory file to a YAML inventory file.
- How to use the **`group_vars`** directory to improve playbook projects maintainability.
- Use special inventory variables to control connection to hosts and make playbook output more readable.

Instructor Tips and Suggestions

Writing YAML inventory files

Writing YAML inventory files

- Many tools now exist to create and parse YAML-formatted files. YAML inventory files allow the integration of other applications that natively work with YAML-formatted files.

  INI to YAML conversion only makes sense if the ansible project is going to be evolved, otherwise, converting just for the sake of it is pointless.

- If students use the **`ansible-inventory`** command to convert inventories, they will likely need to revise the generated output. Group variables are flattened to each host, so there are many duplicative variable definitions. Students should refactor this output to contain group variables in the **`vars`** section of each host group.

Managing Inventory Variables

- When using **`group_vars`** students should understand that Ansible makes a pool of all the variables that a apply for each host, so in the end all variables applied to groups are variables applied to hosts.

Accounting for Training Modalities

When developing course content, you must consider the various training modalities and how your learners will be consuming content. When writing your course, your student guide and instructor guide need to address training delivered in the following training modalities:

Training Modality	Description
Instructor-led Training (ILT)	Traditional, classroom instruction, with in-room networked systems and access to lab environments for each student. Hard copies of the student guide are distributed to each learner. **Note:** All training offerings will be developed as ILTs.
Virtual Training (VT)	Instructor-led, real-time training, delivered virtually (online) using screen-sharing software and featuring a virtual lab environment. Students receive a custom-watermarked e-book student guide.
Red Hat Online Learning (ROLE)	Self-paced online learning, with a virtual lab environment. The student guide is rendered as HTML and presented within the course environment itself.

Writing for VT and ROLE

Once the student guide has been written for ILT, content must be modified in the guide and output as PDF or HTML guides for VT and ROLE offerings. Follow these guidelines when developing for these modalities:

When writing for...	Do the following...	
Virtual Training (VT)	Instructor Guide & Slides	Using the existing instructor guide for ILT, update the following sections: • *Course Timing/Schedule*: Update the schedule for VT times. Consult the Detailed Course Outline for timing of a VT class, placing the VT schedule table below the ILT schedule table. Using the existing student guide for ILT, update the following sections: • *Orientation to the Classroom Environment*: Add the VT instructions and table for connecting to the lab environment over the

	Student Guides (custom-watermarked e-books)	Internet. • *Appendix:* Add instructions on how to interact with Blackboard Collaborate.
Red Hat Online Learning (ROLE)	Student Guide (HTML pages)	Using the existing student guide for ILT, update the following: • *Orientation to the ROL Classroom Environment*: Must be written specifically for ROL virtual lab environment.

Example Orientation to the Classroom section:

Introduction

ORIENTATION TO THE CLASSROOM ENVIRONMENT

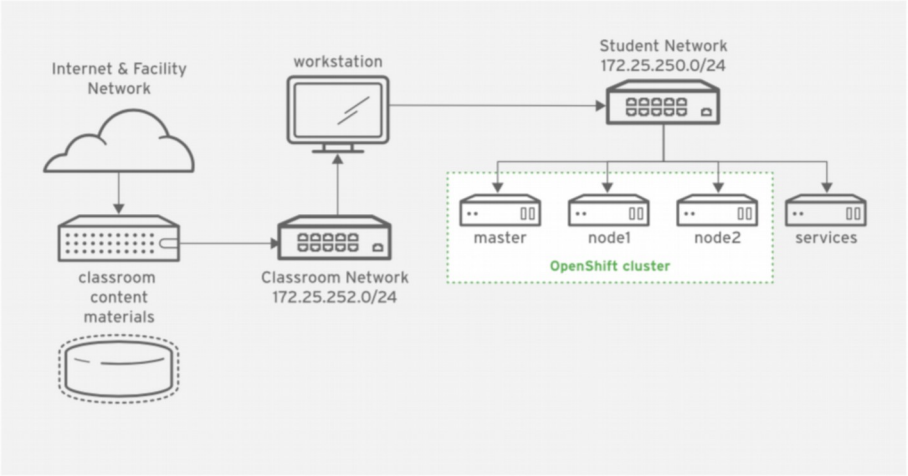


Figure 0.1: Classroom Environment

In this course, the main computer system used for hands-on learning activities is **`workstation`**. Four other machines will also be used by students for these activities. These are **`master`**, **`node1`**, **`node2`**, and **`services`**. All four of these systems are in the **`lab.example.com`** DNS domain.

All student computer systems have a standard user account, **`student`**, which has the password **`student`**. The **`root`** password on all student systems is **`redhat`**.

Classroom Machines

MACHINE NAME	IP ADDRESSES	ROLE
workstation.lab.example.com	172.25.250.254	Graphical workstation used for system administration
master.lab.example.com	172.25.250.10	Master of the OpenShift cluster
node1.lab.example.com	172.25.250.11	Node in the OpenShift cluster
node2.lab.example.com	172.25.250.12	Node in the OpenShift

Tips for Writing Consistently in All Modalities

- Remember that not all students will have the benefit of a live instructor and additional context; illustration or graphical depiction may be necessary.
- Create more digestible chunks of text, rather than long passages (see *Chunking Text* for more information).
- When developing practices and labs, provide sufficient detail in both steps and solutions. Each step must be stated as an action and preparatory information should be listed above steps, rather than as part of the step.

Tagging XML and Adhering to Style and Graphic Standards

The Content Service team has developed the document, GLS DocBook XML Standards If you have additional questions about these guidelines, email David at daobrien@redhat.com.

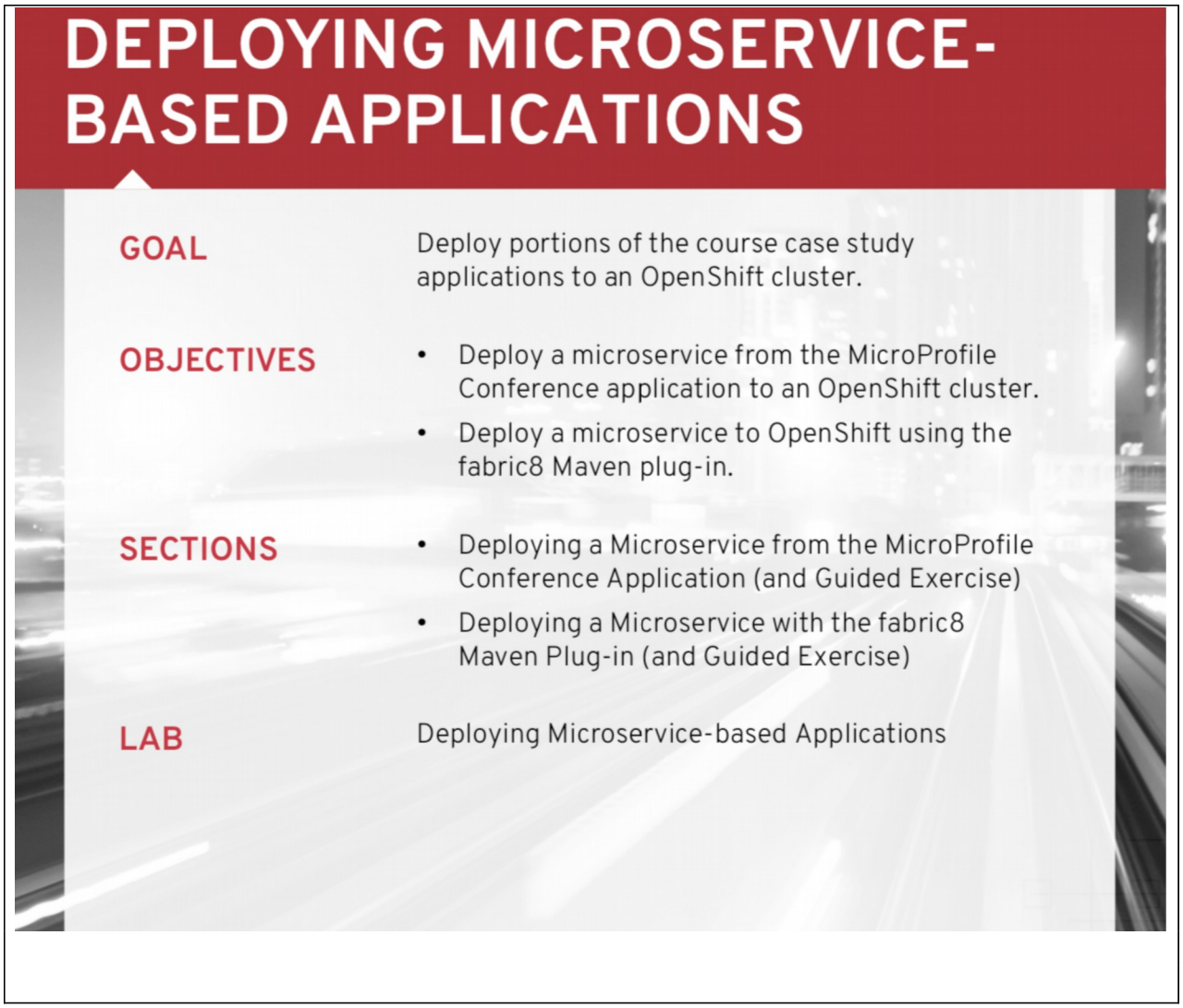
Inserting Introductory Table Content

Insert the chapter goal and objectives in the introductory table. Also, include section titles and the end-of-chapter lab title on the introductory table page. Use the DCO or HLD as your guide to fill in the introductory table. Creating this page adds a valuable guide and quick reference to the chapter content for instructors and students.

Tips for Completing the Introductory Table Page

- Refer to your completed DCO/HLD form to complete the introductory table.
- Write the chapter name starting with a gerund (an -ing verb).
- Write the objective(s) starting with a verb.
- Write the section objectives starting with a verb.
- Write the section titles starting with a gerund (an -ing verb); consider that every chapter is asking the student to do something, even if it's comprehending or being able to identify terms or concepts.
- Add "(and Guided Exercise)" or "(and Quiz)" after each section containing a hands-on practice or quiz.
- Make sure the end-of-chapter lab title matches the chapter title.

Example of a completed introductory table page



Chunking Text

The fastest way to lose student attention is to include large amounts of text on your pages. Breaking content into smaller parts avoids long blocks of text-only content and improves learner retention. It is easier for students to backtrack and find necessary details after they have taken a course if the content is segmented and broken into chunks. Consider using graphics, tables, and lists instead of text or to break up long blocks of text.

Writing with Consistent Grammar and Style

Write GLS training courses using consistent grammar and writing style, as recommended here: *Grammar Sessions and Writer References on CS Mojo*.

If you have additional questions about these guidelines, email the Technical Editors (David) at daobrien@redhat.com.

Writing with Active Voice

Writing in active voice makes your sentences clear, direct, and forceful. Writing directly to your reader commands their attention, improves clarity and helps reduce word count. For all new courses we will be adhering to writing in the active voice. Here are some examples:

Active (Second Person--implied "you")	Passive (Third Person)
Use sudo, or log in as root, to edit the file.	The file can be edited only as root.
Use redundant storage to protect your organization from data loss, and ensure high-availability in your cluster.	Systems administrators will benefit greatly from using redundant storage.
Type <...> to start Linuxconf.	Linuxconf can be started by typing <...>

If you have additional questions about these guidelines, email the Technical Editors (David) at daobrien@redhat.com.

Creating Graphics and Screenshots

Graphics and screenshots add context, specificity, and interest to your course. Use consistent guidelines as recommended on the CS Mojo Graphics page.

Submitting a Graphics Request

Identify graphics as early in the development process as possible. The Graphic Designer should review all graphics, for an updated or new course. You may submit line drawings or freehand sketches for new graphics using the Graphics Request ServiceNow form.

If you have additional questions about these guidelines, email the Graphic Designer (Sajith) at seyamkuz@redhat.com.

Incorporating Tables

When writing your course content, look for repeating text or items that can be grouped within a table. Use a table to create a visual aid for students. For example, if you have a list of five or more commands or definitions, display those in a table rather than as a bulleted list. Tables must include a title and column headings that explain the table contents, and provide a brief description of what you want the students to know about the contents in the table. Bold the table header row. If there are terms in the first column, bold those terms.

Example of a table

Speaker Service Endpoints

ENDPOINT	HTTP METHOD	DESCRIPTION
/speaker	**GET**	Lists all speakers
/speaker/add	**POST**	Adds a speaker
/speaker/remove/{id}	**DELETE**	Removes a speaker by **id**
/speaker/update	**PUT**	Updates an existing speaker
/speaker/retrieve/{id}	**GET**	Retrieves a speaker with a given **id**
/speaker/search	**PUT**	Searches for a speaker

Incorporating Lists

Using lists in your course content emphasizes important points in your instruction and aids in the student's ability to rapidly scan contents of a section. Use bulleted lists in your content for a non-sequential list of items. Use numbered lists for items where order is important. Seven to 10 items should be the maximum number of items in a list. When you are listing items, make sure you write each item in a consistent manner. For example, start every item with a verb, or a gerund, or a noun.

Writing Outcome Statements

Write outcome statements for all guided exercises, labs, and instructor-led labs. Outcomes tell students what they will be able to do after completing a guided lab, lab, or an instructor-led lab. Outcomes restate the essential skill that the student must demonstrate and achieve to complete the guided lab, lab, or instructor-led lab. Begin each outcome with the text, "You should be able to … xxx."

Example outcome: "You should be able to set up SSH user key-based authentication to initiate SSH connections."

Writing Summary Statements

Write a summary statement at the end of every chapter that summarizes the main ideas and key points of the chapter. A summary lists the absolutely critical things the student must know in order to successfully meet the chapter objective. After building your chapter, look back at each section and identify one or two things per section that are critical for the

student to know and retain about the section. A summary statement should *not* restate the section objectives.

Example summary