Course Name

June, 2025

Table of Contents

# About this Course

## 0.1 Available course formats

This course is available in multiple formats which allows you to take it in the way that best suites your needs. You can take it for certificate which can be for free or fee.

* The material for this course can be viewed without login requirement on this [Bookdown website](LINK%20HERE). This format might be most appropriate for you if you rely on screen-reader technology.
* This course can be taken for [free certification through Leanpub](LINK%20HERE).
* This course can be taken on [Coursera for certification here](LINK%20HERE) (but it is not available for free on Coursera).
* Our courses are open source, you can find the [source material for this course on GitHub](LINK%20HERE).

# 1 Assignment 1: Building Your Capstone Dockerfile

## 1.1 **Phase 1: Repository Setup and Planning**

### 1.1.1 **Step 1: Set Up Your Working Environment**

* Clone or navigate to your capstone sandbox repository
* Locate the docker/Dockerfile - this is where you’ll be working
* **Important:** Don’t change the name or location of this file
* Open the existing Dockerfile to see what’s already there

### 1.1.2 **Step 2: Create Your Working Branch**

* Create a new branch for this assignment (e.g., dockerfile-assignment or docker-build)
* Switch to this branch before making any changes
* This keeps your work organized and allows the automated testing to work properly

### 1.1.3 **Step 3: Analyze Your Capstone Project Needs**

* Look at your capstone project files in the repository
* Identify what programming languages you’re using (R, Python, etc.)
* Make a list of packages/libraries your analysis scripts require
* Check if there are any special tools or dependencies needed

## 1.2 **Phase 2: Dockerfile Development**

### 1.2.1 **Step 4: Examine the Base Setup**

* Open docker/Dockerfile and see what base image is already specified
* Read any existing instructions or comments
* Understand what’s already included before adding new components

### 1.2.2 **Step 5: Add Packages Incrementally**

* Start by adding just one or two essential packages to your Dockerfile
* Use the templates from the course:
  + For R packages: RUN Rscript -e "install.packages('packagename')"
  + For Python packages: RUN pip3 install packagename
* Add comments explaining what each package does

### 1.2.3 **Step 6: Build Locally (If Working on Your Computer)**

* If developing locally, test your Dockerfile with:
* cd docker  
  docker build . -t my-capstone-image
* Fix any build errors before pushing to GitHub
* This saves time and GitHub Actions usage

## 1.3 **Phase 3: Testing and Iteration**

### 1.3.1 **Step 7: Open Your Pull Request**

* Commit your Dockerfile changes to your branch
* Push the branch to GitHub
* Open a pull request from your branch to the main branch
* **Key:** This triggers the Docker Assignment Eval test

### 1.3.2 **Step 8: Monitor the Automated Testing**

* Watch for the Docker Assignment Eval check to start running
* This GitHub Action will attempt to build your Dockerfile
* Wait for it to complete and comment on your PR

### 1.3.3 **Step 9: Interpret the Results**

* **Success:** You’ll get a validation code in the comment
* **Failure:** You’ll get error messages explaining what went wrong
* Read the error messages carefully - they contain clues for fixing issues

## 1.4 **Phase 4: Troubleshooting and Refinement**

### 1.4.1 **Step 10: Debug Build Failures**

* **Common issues to check:**
  + Typos in package names
  + Missing dependencies (install system packages before language packages)
  + Incorrect syntax in RUN commands
  + Base image doesn’t support your installation method

### 1.4.2 **Step 11: Apply Troubleshooting Strategies**

* **Package not found:** Check spelling, verify the package exists
* **Installation fails:** Look for missing system dependencies
* **Syntax errors:** Review course templates and examples
* **Memory issues:** Simplify your build, install fewer packages at once

### 1.4.3 **Step 12: Iterate Until Success**

* Make changes to your Dockerfile on your branch
* Commit and push changes
* Each push will trigger a new test run
* Keep refining until you get a successful build

## 1.5 **Phase 5: Completion**

### 1.5.1 **Step 13: Collect Your Validation Code**

* Once your build succeeds, copy the validation code from the PR comment
* **Important:** Make sure it’s a real validation code, not an error message
* Keep this code safe - you’ll need it for your quiz

### 1.5.2 **Step 14: Submit Your Quiz**

* Go to your Coursera Capstone Quiz
* Paste the validation code when prompted
* Submit the quiz to get credit for your work

## 1.6 **Pro Tips for Success**

### 1.6.1 **Start Simple**

* Begin with just the most essential packages for your analysis
* You can always add more in subsequent iterations
* A working simple image is better than a broken complex one

### 1.6.2 **Use the Course Resources**

* Reference the package installation templates from the Containers for Scientists course
* Look for examples of other people installing similar packages
* Don’t reinvent the wheel

### 1.6.3 **Leverage the Automated Testing**

* The GitHub Action is your friend - it provides immediate feedback
* Don’t be afraid to make multiple attempts
* Each failure teaches you something about container building

### 1.6.4 **Read Error Messages Carefully**

* Error messages often contain the exact solution
* Look for phrases like “package not found” or “missing dependency”
* Google specific error messages if they’re unclear

### 1.6.5 **Document Your Process**

* Add comments to your Dockerfile explaining your choices
* This helps you remember your reasoning if you need to modify it later
* Good documentation makes debugging easier

Remember: This assignment is designed to give you hands-on experience with the container building process. The automated testing system provides immediate feedback, making it easier to learn through iteration rather than getting everything perfect on the first try.

# 2 Assignment 2: Building Your Capstone GitHub Action

## 2.1 **Phase 1: Repository Setup and Planning**

### 2.1.1 **Step 1: Set Up Your Working Environment**

* Navigate to your capstone sandbox repository
* Locate the ASSIGNMENT\_2.yml file - this is your starting template
* **Important:** Keep the filename exactly as ASSIGNMENT\_2.yml throughout the assignment
* Review the existing template to understand what’s already provided

### 2.1.2 **Step 2: Create Your Working Branch**

* Create a new branch for this assignment (e.g., github-action-assignment or gha-build)
* Switch to this branch before making any changes
* This isolates your work and enables the automated evaluation system

### 2.1.3 **Step 3: Plan Your GitHub Action**

* Decide what useful task your GitHub Action will perform
* **Ideas for useful actions:**
  + Run automated tests on your capstone code
  + Generate reports or documentation
  + Check code quality or formatting
  + Create data visualizations
  + Send notifications when certain conditions are met
  + Validate data files or configurations

## 2.2 **Phase 2: GitHub Action Development**

### 2.2.1 **Step 4: Understand the Template Structure**

* Open ASSIGNMENT\_2.yml and examine the existing structure
* Identify the key components:
  + name: - What your action is called
  + on: - When it should trigger
  + jobs: - What it should do
  + runs-on: - What environment to use

### 2.2.2 **Step 5: Move File to Correct Location**

* GitHub Actions must be in the .github/workflows/ directory
* Move ASSIGNMENT\_2.yml to .github/workflows/ASSIGNMENT\_2.yml
* **Critical:** The file must be in this exact location for GitHub to recognize it

### 2.2.3 **Step 6: Define Your Action’s Purpose**

* **Choose a meaningful trigger (on:):**
  + pull\_request: - Runs when PRs are opened/updated (good for testing)
  + push: - Runs when code is pushed to specific branches
  + workflow\_dispatch: - Allows manual triggering (useful for development)
  + schedule: - Runs on a time schedule
* **Design your job steps:**
  + Start with actions/checkout@v4 to get your repository files
  + Add steps that accomplish your chosen task
  + Include error handling and status checks

## 2.3 **Phase 3: Implementation Strategies**

### 2.3.1 **Step 7: Start Simple and Build Up**

* Begin with a basic action that you know will work
* Test early and often to catch issues quickly
* Add complexity gradually, testing each addition

### 2.3.2 **Step 8: Use Course Knowledge**

* **Apply containers knowledge:**
  + Choose appropriate runs-on: environment
  + Consider using Docker containers if you need specific software
  + Reference container images from Docker Hub if needed
* **Apply automation principles:**
  + Ensure your action fails appropriately when something goes wrong
  + Include meaningful output and logging
  + Use environment variables and secrets when needed

### 2.3.3 **Step 9: Common Implementation Patterns**

#### 2.3.3.1 **For Code Quality/Testing:**

name: Code Quality Check  
on: pull\_request  
jobs:  
 quality-check:  
 runs-on: ubuntu-latest  
 steps:  
 - uses: actions/checkout@v4  
 - name: Run quality checks  
 run: |  
 # Your quality check commands here  
 echo "Running code quality checks..."

#### 2.3.3.2 **For Data Analysis:**

name: Automated Analysis  
on: workflow\_dispatch  
jobs:  
 analyze:  
 runs-on: ubuntu-latest  
 container:  
 image: jhudsl/ottr\_python:main # Example with R and Python  
 steps:  
 - uses: actions/checkout@v4  
 - name: Run analysis  
 run: |  
 # Your analysis commands here

#### 2.3.3.3 **For Documentation:**

name: Generate Documentation  
on: push  
jobs:  
 docs:  
 runs-on: ubuntu-latest  
 steps:  
 - uses: actions/checkout@v4  
 - name: Generate docs  
 run: |  
 # Documentation generation commands

## 2.4 **Phase 4: Testing and Iteration**

### 2.4.1 **Step 10: Open Your Pull Request**

* Commit your changes to your branch
* Push the branch to GitHub
* Open a pull request from your branch to main
* **Key:** This triggers the GHA Assignment Eval test

### 2.4.2 **Step 11: Monitor Automated Evaluation**

* Watch for the GHA Assignment Eval check to start running
* This evaluator will test whether your GitHub Action runs successfully
* Wait for it to complete and comment on your PR

### 2.4.3 **Step 12: Interpret Evaluation Results**

* **Success:** You’ll receive a validation code in the PR comment
* **Failure:** You’ll get specific error messages and troubleshooting tips
* The evaluator checks for common issues and provides guidance

## 2.5 **Phase 5: Troubleshooting and Refinement**

### 2.5.1 **Step 13: Debug Common Issues**

* **YAML syntax errors:** Check indentation, colons, and spacing
* **File location errors:** Ensure file is in .github/workflows/
* **Permission errors:** May need to add GitHub secrets or tokens
* **Missing dependencies:** Check if your chosen environment has required software
* **Silent failures:** Verify your action actually does what you expect

### 2.5.2 **Step 14: Apply Troubleshooting Strategies**

* **Read error messages carefully:** They often contain the exact solution
* **Check the logs:** Go to Actions tab and examine detailed output
* **Test incrementally:** Make small changes and test each one
* **Use marketplace actions:** Leverage existing actions when possible
* **Print debugging info:** Use echo commands to verify assumptions

### 2.5.3 **Step 15: Iterate Until Success**

* Make changes to your ASSIGNMENT\_2.yml file on your branch
* Commit and push changes to trigger new evaluation runs
* Keep refining based on feedback until you get a successful build

## 2.6 **Phase 6: Completion**

### 2.6.1 **Step 16: Collect Your Validation Code**

* Once your GitHub Action runs successfully, copy the validation code from the PR comment
* **Verify:** Make sure it’s a real validation code, not an error message
* Keep this code safe for your quiz submission

### 2.6.2 **Step 17: Submit Your Quiz**

* Navigate to your Coursera Capstone Quiz
* Paste the validation code when prompted
* Submit the quiz to receive credit for your work

## 2.7 **Pro Tips for Success**

### 2.7.1 **Choose Appropriate Scope**

* Your action should be useful but not overly complex
* Focus on demonstrating GitHub Actions concepts rather than building production software
* A working simple action is better than a broken complex one

### 2.7.2 **Leverage Course Materials**

* Use troubleshooting strategies from the GitHub Actions course
* Reference YAML examples from course exercises
* Apply container knowledge if your action needs specific software environments

### 2.7.3 **Use the Evaluation System Effectively**

* The automated evaluator provides immediate feedback
* Don’t be afraid to make multiple attempts - each iteration teaches you something
* Read the evaluator’s comments carefully for specific guidance

### 2.7.4 **Think Like a Developer**

* Consider when your action should run (triggers)
* Think about error handling and edge cases
* Include meaningful logging and output messages
* Test your assumptions about what the action environment provides

### 2.7.5 **Common Useful Actions for Capstone Projects**

* **Data validation:** Check that uploaded data meets expected formats
* **Report generation:** Automatically create summary reports from data
* **Code style checking:** Ensure consistent formatting and style
* **Dependency checking:** Verify all required packages are available
* **Backup creation:** Automatically backup important files
* **Notification system:** Send alerts when certain conditions are met

Remember: This assignment builds on both the containers knowledge (for choosing appropriate environments) and the GitHub Actions automation concepts. The goal is to demonstrate your understanding of continuous integration principles by creating something that automatically improves or validates your capstone project.

The automated evaluation system is designed to help you learn through iteration, so embrace the feedback loop and keep refining your action until it works reliably!

# About the Authors

These credits are based on our [course contributors table guidelines](https://www.ottrproject.org/more_features.html#giving-credits-to-contributors).

| Credits | Names |
| --- | --- |
| **Pedagogy** |  |
| Lead Content Instructor(s) | [FirstName LastName](link%20to%20personal%20website) |
| Lecturer(s) (include chapter name/link in parentheses if only for specific chapters) - make new line if more than one chapter involved | Delivered the course in some way - video or audio |
| Content Author(s) (include chapter name/link in parentheses if only for specific chapters) - make new line if more than one chapter involved | If any other authors besides lead instructor |
| Content Contributor(s) (include section name/link in parentheses) - make new line if more than one section involved | Wrote less than a chapter |
| Content Editor(s)/Reviewer(s) | Checked your content |
| Content Director(s) | Helped guide the content direction |
| Content Consultants (include chapter name/link in parentheses or word “General”) - make new line if more than one chapter involved | Gave high level advice on content |
| Acknowledgments | Gave small assistance to content but not to the level of consulting |
| **Production** |  |
| Content Publisher(s) | Helped with publishing platform |
| Content Publishing Reviewer(s) | Reviewed overall content and aesthetics on publishing platform |
| **Technical** |  |
| Course Publishing Engineer(s) | Helped with the code for the technical aspects related to the specific course generation |
| Template Publishing Engineers | [Candace Savonen](https://www.cansavvy.com/), [Carrie Wright](https://carriewright11.github.io/), [Ava Hoffman](https://www.avahoffman.com/) |
| Publishing Maintenance Engineer | [Candace Savonen](https://www.cansavvy.com/) |
| Technical Publishing Stylists | [Carrie Wright](https://carriewright11.github.io/), [Ava Hoffman](https://www.avahoffman.com/), [Candace Savonen](https://www.cansavvy.com/) |
| Package Developers ([ottrpal](https://github.com/jhudsl/ottrpal)) [Candace Savonen](https://www.cansavvy.com/), [Ava Hoffman](https://www.avahoffman.com/), [Howard Baek](https://www.linkedin.com/in/howard-baik/), [Kate Isaac](https://kweav.github.io/), [Carrie Wright](https://carriewright11.github.io/), [John Muschelli](https://johnmuschelli.com/) |  |
| **Art and Design** |  |
| Illustrator(s) | Created graphics for the course |
| Figure Artist(s) | Created figures/plots for course |
| Videographer(s) | Filmed videos |
| Videography Editor(s) | Edited film |
| Audiographer(s) | Recorded audio |
| Audiography Editor(s) | Edited audio recordings |
| **Funding** |  |
| Funder(s) | Institution/individual who funded course including grant number |
| Funding Staff | Staff members who help with funding |

## ─ Session info ───────────────────────────────────────────────────────────────  
## setting value  
## version R version 4.3.2 (2023-10-31)  
## os Ubuntu 22.04.4 LTS  
## system x86\_64, linux-gnu  
## ui X11  
## language (EN)  
## collate en\_US.UTF-8  
## ctype en\_US.UTF-8  
## tz Etc/UTC  
## date 2025-06-06  
## pandoc 3.1.1 @ /usr/local/bin/ (via rmarkdown)  
##   
## ─ Packages ───────────────────────────────────────────────────────────────────  
## package \* version date (UTC) lib source  
## bookdown 0.41 2024-10-16 [1] CRAN (R 4.3.2)  
## cachem 1.0.8 2023-05-01 [1] RSPM (R 4.3.0)  
## cli 3.6.2 2023-12-11 [1] RSPM (R 4.3.0)  
## devtools 2.4.5 2022-10-11 [1] RSPM (R 4.3.0)  
## digest 0.6.34 2024-01-11 [1] RSPM (R 4.3.0)  
## ellipsis 0.3.2 2021-04-29 [1] RSPM (R 4.3.0)  
## evaluate 0.23 2023-11-01 [1] RSPM (R 4.3.0)  
## fastmap 1.1.1 2023-02-24 [1] RSPM (R 4.3.0)  
## fs 1.6.3 2023-07-20 [1] RSPM (R 4.3.0)  
## glue 1.7.0 2024-01-09 [1] RSPM (R 4.3.0)  
## htmltools 0.5.7 2023-11-03 [1] RSPM (R 4.3.0)  
## htmlwidgets 1.6.4 2023-12-06 [1] RSPM (R 4.3.0)  
## httpuv 1.6.14 2024-01-26 [1] RSPM (R 4.3.0)  
## knitr 1.48 2024-07-07 [1] CRAN (R 4.3.2)  
## later 1.3.2 2023-12-06 [1] RSPM (R 4.3.0)  
## lifecycle 1.0.4 2023-11-07 [1] RSPM (R 4.3.0)  
## magrittr 2.0.3 2022-03-30 [1] RSPM (R 4.3.0)  
## memoise 2.0.1 2021-11-26 [1] RSPM (R 4.3.0)  
## mime 0.12 2021-09-28 [1] RSPM (R 4.3.0)  
## miniUI 0.1.1.1 2018-05-18 [1] RSPM (R 4.3.0)  
## pkgbuild 1.4.3 2023-12-10 [1] RSPM (R 4.3.0)  
## pkgload 1.3.4 2024-01-16 [1] RSPM (R 4.3.0)  
## profvis 0.3.8 2023-05-02 [1] RSPM (R 4.3.0)  
## promises 1.2.1 2023-08-10 [1] RSPM (R 4.3.0)  
## purrr 1.0.2 2023-08-10 [1] RSPM (R 4.3.0)  
## R6 2.5.1 2021-08-19 [1] RSPM (R 4.3.0)  
## Rcpp 1.0.12 2024-01-09 [1] RSPM (R 4.3.0)  
## remotes 2.4.2.1 2023-07-18 [1] RSPM (R 4.3.0)  
## rlang 1.1.4 2024-06-04 [1] CRAN (R 4.3.2)  
## rmarkdown 2.25 2023-09-18 [1] RSPM (R 4.3.0)  
## sessioninfo 1.2.2 2021-12-06 [1] RSPM (R 4.3.0)  
## shiny 1.8.0 2023-11-17 [1] RSPM (R 4.3.0)  
## stringi 1.8.3 2023-12-11 [1] RSPM (R 4.3.0)  
## stringr 1.5.1 2023-11-14 [1] RSPM (R 4.3.0)  
## urlchecker 1.0.1 2021-11-30 [1] RSPM (R 4.3.0)  
## usethis 2.2.3 2024-02-19 [1] RSPM (R 4.3.0)  
## vctrs 0.6.5 2023-12-01 [1] RSPM (R 4.3.0)  
## xfun 0.48 2024-10-03 [1] CRAN (R 4.3.2)  
## xtable 1.8-4 2019-04-21 [1] RSPM (R 4.3.0)  
## yaml 2.3.8 2023-12-11 [1] RSPM (R 4.3.0)  
##   
## [1] /usr/local/lib/R/site-library  
## [2] /usr/local/lib/R/library  
##   
## ──────────────────────────────────────────────────────────────────────────────

# 3 References