

Lecture Notes Spring 2020

Cloud Computing

Engineering

e516

Gregor von Laszewski

Editor

laszewski@gmail.com

<https://cloudmesh-community.github.io/pub/vonlaszewski-e516.epub>

January 23, 2020 - 06:36 PM

Created by Cloudmesh & Cyberaide Bookmanager, <https://github.com/cyberaide/bookmanager>

E516 - ENGINEERING CLOUD COMPUTING

Gregor von Laszewski

(c) Gregor von Laszewski, 2020

E516 - ENGINEERING CLOUD COMPUTING

1 PREFACE

1.1 Disclaimer ☁

1.1.1 Acknowledgment

1.1.2 Extensions

2 SYLLABUS

2.0.1 Syllabus Engineering Cloud Computing ☁

2.0.1.1 Instructor

2.0.1.2 Audience

2.0.1.3 Course summary

2.0.1.4 References

2.0.1.5 Tools

2.0.1.6 Course Schedule

2.0.1.7 Attendance

2.0.1.8 Assignments

2.0.1.9 Grade Breakdown

2.0.1.10 Project Examples

2.0.1.11 Statement on Academic Misconduct

2.1 Calendar ☁

3 OVERVIEW

3.1 Summary ☁

3.2 Communication ☁

3.2.1 Class Material

3.2.2 Piazza

3.2.3 Class Resources

3.2.4 Online Meeting

3.2.5 Assignments

3.2.6 Post Your Bio

3.2.7 Evolving Document

3.2.8 Books




3.2.9 You Get Credit for Improving the Books






3.2.10 Ongoing Agenda

3.3 Quick Tips for the Class ☁

- [3.3.1 Requirements](#)
- [3.3.2 Time Commitment](#)
- [3.3.3 Course Material List](#)
- [3.3.4 Help](#)
- [3.3.5 How to Take this Class](#)
- [3.3.6 Assignments](#)
 - [3.3.6.1 Technology Review](#)
 - [3.3.6.2 Project](#)
 - [3.3.6.2.1 License](#)
 - [3.3.6.2.2 Project Report](#)
 - [3.3.6.2.3 Project Code](#)
 - [3.3.6.2.4 Project Data](#)
 - [3.3.6.2.5 Work Breakdown](#)
 - [3.3.6.2.6 Bibliography](#)
 - [3.3.6.2.7 Reproducibility](#)
 - [3.3.6.2.8 List of Deliverables](#)
 - [3.3.6.2.9 Example Outline of a Report](#)
- [3.3.7 Submission](#)
- [3.3.8 Bonus Projects](#)
- [3.3.9 Participation](#)
- [3.3.10 GitHub and Pycharm Markdown](#)

[4 WEEKLY AGENDA](#)

- [4.1 Week 1: Course Introduction](#) 
 - [4.1.1 Assignments](#)
 - [4.1.2 Class Videos](#)
 - [4.1.3 Lab Activities](#)
 - [4.1.3.1 Account Creation](#)
 - [4.1.3.2 Survey](#)
 - [4.1.3.3 Bio](#)
 - [4.1.3.4 Python 3.8](#)
 - [4.1.4 Online Meeting Recording](#) 
- [4.2 Week 2: Cloud Data Centers](#) 
 - [4.2.1 Lecture Material](#)
 - [4.2.2 Lab Activities](#)
 - [4.2.2.1 Receive your HID](#)
 - [4.2.2.2 README.yml](#)

- [4.2.2.3 Chameleon Cloud](#)
 - [4.2.2.4 Review Python](#)
 - [4.2.2.5 Setup your Computer](#)
 - [4.2.2.5.1 Python](#)
 - [4.2.2.6 Ubuntu Multipass](#)
 - [4.2.2.7 Optional: Plagiarism Certificate](#)
 - [4.2.3 Online Meeting Recording](#) 
 - [4.3 Week 3: Cloud Computing Architectures](#)  
 - [4.3.1 Lecture Material](#)
 - [4.3.2 Lab Activities](#)
 - [4.3.2.1 Review: venv in python 3 \(Graded\)](#)
 - [4.3.2.2 Review: Anaconda \(Graded\)](#)
 - [4.3.2.2.1 Dicts](#)
 - [4.3.2.2.2 f-Strings in Python 3](#)
 - [4.3.2.2.3 Classes](#)
 - [4.3.2.2.4 Python Modules](#)
 - [4.3.2.3 Graded Activities](#)
 - [4.3.3 Chapter Selection](#)
 - [4.3.4 Project Selection](#)
 - [4.3.5 Working Ahead](#)
 - [4.3.6 Online Meeting Recording](#)
 - [4.4 Week 4: Openstack](#)  
 - [4.4.1 Video](#)
 - [4.4.2 Lecture Material](#)
 - [4.4.3 Lab: OpenStack](#)
 - [4.4.4 Naming of vms](#)
 - [4.4.5 Horizon](#)
 - [4.4.6 Cloudmesh OpenStack interface](#)
 - [4.4.7 OpenStack Command line Client](#)
 - [4.4.7.1 Installation of Cloudmesh Cloud Bundle](#)
 - [4.4.7.2 SSH](#)
 - [4.4.7.3 Configuration](#)
 - [4.4.7.4 Cloudmesh Mongo](#)
 - [4.4.7.5 Start a VM](#)
 - [4.4.8 Working with the VM](#)
 - [4.4.9 Exercise](#)
-

4.5 Week 5: Cloud VM Compute Service

4.5.1 Videos

4.5.2 Lecture Material

4.5.3 Lab: VM Compute Services

4.5.4 Exercise

4.6 Week 6: REST

4.6.1 Lecture Material

4.6.2 Lab Activities

4.6.3 Graded Lab Activity

4.7 Week 7: MapReduce

4.7.1 Lecture Material

4.7.2 Lab Activity: Hadoop Installation

4.7.2.1 Exercises:

4.7.3 Lab Activity: Python MapReduce

4.7.4 Lab Activity Map/Reduce on the Cloud

4.7.4.1 Python Word Count in MapReduce

4.7.4.2 Run Hadoop MapReduce in Python

4.7.5 Lab Activity: MapReduce on the cloud

4.7.5.1 Hadoop Cluster Setup

4.7.5.2 Configuration

4.7.5.3 Operating the Hadoop Cluster

4.7.5.4 Run Hadoop MapReduce in Python

4.8 Week 8: Project Review

4.9 Week 9: Containers - Docker

4.9.1 Lecture Material

4.9.2 Lab Activity Docker

4.9.3 Lab Activity Container REST service

4.10 Week 11: GraphQL and Messaging

4.10.1 Lecture Material

4.10.2 Lab Activity GraphQL

4.10.3 Optional Lab Activity MQTT

4.11 Week 12: GO

4.11.1 Optional Lab Activity Go REST Service

4.12 Week 13: FaaS

4.12.1 Lab Activity

4.13 Week N - 16

5 APPENDIX

5.1 FAQs ☁

5.1.1 Where do I find the material for this class?

5.1.2 Where to find the weekly agenda?

5.1.3 When do the online meetings take place?

5.1.4 Where can I find the recordings of the online meetings?

5.1.5 Do I need to read the class material to take the class?

5.1.6 Lab: Why can I not solve an assignment in 2 minutes?

5.1.7 Lab: How much time do I need to spend on each question?

5.1.8 I do not understand Piazza?

5.1.9 Why you should not read the Piazza mail but use [click Here](#) instead?

5.1.10 Can I search in the ePub reader?


5.1.11 Where do I find the GitHub directory?

5.1.12 Where can I find the hid directory?

5.2 ePub Readers ☁

6 REFERENCES

1 PREFACE

Thu Jan 23 18:35:59 EST 2020 

1.1 DISCLAIMER

This book has been generated with [Cyberaide Bookmanager](#).

Bookmanager is a tool to create a publication from a number of sources on the internet. It is especially useful to create customized books, lecture notes, or handouts. Content is best integrated in markdown format as it is very fast to produce the output.

Bookmanager has been developed based on our experience over the last 3 years with a more sophisticated approach. Bookmanager takes the lessons from this approach and distributes a tool that can easily be used by others.

The following shields provide some information about it. Feel free to click on them.



1.1.1 ACKNOWLEDGMENT

If you use bookmanager to produce a document you must include the following acknowledgement.

“This document was produced with Cyberaide Bookmanager developed by Gregor von Laszewski available at <https://pypi.python.org/pypi/cyberaide-bookmanager>. It is in the responsibility of the user to make sure an author acknowledgement section is included in your document. Copyright verification of content included in a book is responsibility of the book editor.”

The bibtex entry is

```
@Misc{www-cyberaide-bookmanager,  
  author = {Gregor von Laszewski},  
  title = {{Cyberaide Book Manager}},  
  howpublished = {pypi},  
  month = apr,  
  year = 2019,  
  url={https://pypi.org/project/cyberaide-bookmanager/}  
}
```

1.1.2 EXTENSIONS

We are happy to discuss with you bugs, issues and ideas for enhancements. Please use the convenient github issues at

- <https://github.com/cyberaide/bookmanager/issues>

Please do not file with us issues that relate to an editors book. They will provide you with their own mechanism on how to correct their content.

2 SYLLABUS

2.0.1 SYLLABUS ENGINEERING CLOUD COMPUTING



Learning Objectives

- Get a quick overview of what the class is about

2.0.1.1 Instructor

- Gregor von Laszewski laszewski@gmail.com
- Office hours: By appointment

2.0.1.2 Audience

- We recommend you know one programming language.
- Knowledge of python is of advantage but not required. Python is easy. All assignments and projects are conducted in Python.

2.0.1.3 Course summary

This class introduces you to state-of-the-art cloud computing concepts and engineering approaches. This will include virtual machines, containers, and Map/Reduce. The course has Labs in which you can practically explore these concepts. You will, for example, have the option to create a cloud as part of this course and explore cloud computing tools and frameworks.

2.0.1.4 References

The course does not have a required reading list. We provide the following references as starting pointers to what we discuss:

- [Cloud Computing](#)
- [Linux](#)
- [Python](#)
- [Markdown](#)

However, please remember we will select topics and include material that is or may not be covered in these books. The books will evolve during the semester.

2.0.1.5 Tools

You are required to have a computer to log into the cloud. We give you access to an OpenStack cloud. Access to Azure, AWS, Google and others can be achieved through their free tier.

2.0.1.6 Course Schedule

Week	References
1	Course Introduction
2	Cloud Data Centers
3	Python for Cloud Computing, Start of Project Selection
4	Cloud Architectures
5	Virtualization I - OpenStack
6	Virtualization II - AWS, Azure, Google
7	Multi-Cloud Environment
8	Cloud Technology Presentation - Project Review
9	Containers - Docker, Kubernetes
10	Map Reduce
11	Messaging
12	REST
13	GO

Week	References
14	Project Work
15	Projects Due
16	Project Improvements

For each of the topics, you will find one or more relevant chapters or sections in our online book.

2.0.1.7 Attendance

Attendance accounts for 10% of your final grade. If you need to skip class for any reason, you need to notify the instructor and TAs.

2.0.1.8 Assignments

This course will not have exams. Instead, we have the following graded assignment categories:

- *Lab Assignments (pass/fail)*: are assignments that will be conducted on a weekly basis. They will help you make sure you not only understand the material theoretically but try them out.
- *Cloud Technology Chapter with Examples (Graded)*: (This can be substituted if your project is related to Cloudmesh or NIST while providing a manual and usage example). This is a document about a Cloud technology that is not yet included in our books to introduce an interested party to it. It should not contain advertisements but be a rational description of the technology with examples that you have tried yourself. You will have to give a non-plagiarized presentation and document about it. The document is formulated as a chapter to the book with proper references and working examples. Due: Midterm
- *Project Assignments (Graded)*: The most important part of the class for which you will be working throughout the semester. Up to three

students can work in a project. In case of group projects, the project deliverables are increased. Due: after midterm every week submissions.

To support the project evolution, it has three submissions that are spread throughout the semester. Each submission builds on the previous one and modifies previous documents into a consistent project report and documentation for your project.

- Project Outline

A description of what your project is about and how it relates to cloud computing and address:

- What is the problem you try to address?
- What are you doing to address this problem?
- How are you addressing this problem?
- What is the architecture that addresses the problem that you will implement?

- Code and Documentation Review

- You will be asked to have a meeting with the TA's and/or instructor to showcase your code and have at least one review prior to your final submission.
- This will take usually take place through the Lab hours on regular basis.
- A first project discussion must have been done at least once at midterm time.

- Final Project Submission

- All code and documentation must be checked into GitHub well before the semester is over. This allows us to give you feedback for improvements.

In addition to these major milestones, you will have a project discussion and update to report on every week starting from the midterm date.

Please note that the syllabus is subject to change. We may also change the order. Students in this class often come from a wide variety of backgrounds and experiences. As such, the instructor reserves the right to change the content of the course to accommodate the needs and expectations of the students.

2.0.1.9 Grade Breakdown

- 70% Project
- 10% Chapter
- 10% Assignments
- 10% Participation

There is no written exam for the midterm or final in this class. However, there will be a project review at midterm and final time. During the last two weeks of the semester, intensive reviews are conducted. Please do not assume that after you have handed in your project, you are done. You have the ability to improve it based on feedback.

The deliverables for midterm and project reviews are discussed elsewhere

2.0.1.10 Project Examples

A small list of project examples are available at:

- <https://cloudmesh.github.io/cloudmesh-manual/projects/index.html>

We will add to this list as the semester progresses. We like you to identify a project within the first month of the semester.

2.0.1.11 Statement on Academic Misconduct

Students will be expected to uphold and maintain academic and professional honesty and integrity as outlined in the Code of Student Rights, Responsibilities, and Conduct. Cases of academic misconduct will be handled according to the student disciplinary procedures described in IU's policies.

2.1 CALENDAR

Please alert us if we made a mistake!

Here we note the IU calendar and map the week numbers. Please note that the spring break week is not included in the numbers. We strongly suggest you plan a vacation wisely. Often Spring break is a good time to catch up or work ahead.

Spring 2020

See <https://registrar.indiana.edu/official-calendar/official-calendar-spring.shtml> for the official calendar.

- Begins Mon, Jan 13
- MLK Jr. Day Mon, Jan 20
- Auto W Sun, Mar 15
- Spring Break Mar 15 - Mar 22
- Final Exams May 4 - May 8
- Ends Fri, May 8



This list will be updated throughout the semester.

Week	Date	Graded Assignments
1	13 Jan - 19 Jan	Piazza, Bio, Survey
2	20 Jan - 26 Jan	Python3.8, Multipass, HID, github

Week	Date	Graded Assignments
3	27 Jan - 2 Feb	Technology Chapter and Project Update
4	3 Feb - 9 Feb	Technology Chapter and Project Update
5	10 Feb - 16 Feb	Technology Chapter and Project Update
6	17 Feb - 23 Feb	Project Proposal
7	24 Feb - 1 Mar	Project Update
8	2 Mar - 8 Mar	Project Update
9	9 Mar - 15 Mar	Project Update
-	16 Mar - 22 Mar	Break, good time to work ahead
10	23 Mar - 29 Mar	Project Update
11	30 Mar - 5 Apr	Project Update
12	6 Apr - 12 Apr	Project Update
13	13 Apr - 19 Apr	Project Update
14	20 Apr - 26 Apr	Project Due
15	27 Apr - 3 May	Project Improvements
16	3 May - 8 May	Project Improvements

In addition to the listed **Graded Assignments** Labs will be posted. All activities will be listed in the weekly updated Lecture Notes so you can identify what has to be done.

Please consult the weekly lecture notes for the complete list of activities.

Deadlines:

- Feb 3: Last day to propose the chapter
- Feb 10: Last day to submit the first version of the chapter
- Feb 10: Last day to submit your project proposal
- Feb 17: Last day to submit your peer review of another students chapter
- Feb 24: Last day to improve your own chapter

Planned Deaddlines:

- Mar 8: Last day to submit your midterm project report
- Apr 26: First Project review of final project
- May 3: opportunity to improve
- Projects that have passed the reviews on the 26 and 3 have additional days to complete

3 OVERVIEW

3.1 SUMMARY

Please watch the following video and read this chapter.

-  [Overview Engineering Cloud Computing](#)

You may like some of the quotes from students:

“Best class I ever took at IU.”

“Thanks to you I got a job at top rated company.”

“My company has adopted lessons learned from your class.”

“The level of support and expertise the professor provides is outstanding and unmatched within IU.”

3.2 COMMUNICATION

3.2.1 CLASS MATERIAL

Most class material is distributed as ePubs and PDF or resources on the Web. Our ePubs render on many computers very nicely. Thus we recommend you use ePub instead of the PDF versions of our documents. Please install an ePub Reader if your system does not have one.

- macOS: has a built ePub reader called Books.
- Windows 10: [Calibre](#)
- Linux: [Calibre](#)
- Others: Linux: [Calibre](#)

Naturally, you can read the ePubs also on your tablets instead of on your computers, if you have a tablet with lots of memory.

Please be aware that the documents will be updated weekly, and you need to download them accordingly. Make sure you clear the cache if your browser prevents the download of the newest version.

3.2.2 PIAZZA

This class uses Piazza for communication. It is your responsibility to enroll in the Piazza for the class. A link to our Piazza is provided in CANVAS. Piazza works just like a forum in which you can ask questions or post notes. In case of a question, students and teachers can formulate answers. A thread system in the question can be used to gather the answer.

- <https://piazza.com/iu/spring2020/e516spring2020/resources>

You will get a grade for participation in the class discussions on Piazza. Please note that we do not recommend you go from CANVAS to Piazza as CANVAS has an bad HTML layout and in fact, does not allow you to see the Piazza hyperlinks. Instead, go to Piazza directly. Use our link.

3.2.3 CLASS RESOURCES

All class material will be posted at

- <https://piazza.com/iu/spring2020/e516spring2020/resources>

3.2.4 ONLINE MEETING

If you are an online student:

A poll has been posted that you must fill out before Friday of the first week of the semester for possible meeting times.

- <https://piazza.com/class/k4xfd8nxutk4ka?cid=9>

The link to the meeting information is at

- <https://piazza.com/class/k4xfd8nxutk4ka?cid=7>

3.2.5 ASSIGNMENTS

All assignments will be posted in our lecture notes. Graded assignments will also appear in CANVAS. You can find the link to them in our resource section:

- <https://piazza.com/iu/spring2020/e516spring2020/resources>

3.2.6 POST YOUR BIO

To assure that you have access to Piazza and can post to it, please, post a formal Bio to:

- <https://piazza.com/class/k4xfd8nxutk4ka>

Please investigate what a formal Bio is.

3.2.7 EVOLVING DOCUMENT

This document is improved throughout the semester with weekly activities. We recommend that you regularly download a new version. Be aware that some browsers or ePub readers may cache the previous version. Thus make sure to check out the newest version.

3.2.8 BOOKS

This class does not use books from publishers as cloud computing is constantly changing, and by the time you place the purchase order, the book may already be outdated. Instead, we have prepared online books that are continually evolving. It is even expected that tools or services you use will evolve during the semester.

We have the following books in ePub and PDF available. The ePub version is our preferred version:

- [List of Books](#)

The weekly Lecture notes are contained at

- [e516: Cloud Computing Engineering](#)


This page will bring you to the following books that we use as part of e516. It will also clearly specify which assignments you need to do as part of the weekly Lab activities.

- [Cloud Computing](#)
- [Linux](#)
- [Python](#)
- [Markdown](#)

3.2.9 YOU GET CREDIT FOR IMPROVING THE BOOKS

Please note that improving the document in GitHub via pull requests will get you credits. This is an easy opportunity for you to get an excellent grade.

3.2.10 ONGOING AGENDA

 We will add here weekly the topics and Labs that we will cover in the coming week(s). Please be aware that just as in regular lectures, we add the agenda items and follow the syllabus. We may change the order of the topics covered in the syllabus as appropriate.

3.3 QUICK TIPS FOR THE CLASS

3.3.1 REQUIREMENTS

We recommend that you know one programming language. Although most activities are done in Python, this programming language does not have to be Python, as it can easily and quickly be learned throughout the semester. No background in cloud computing is needed.

3.3.2 TIME COMMITMENT

Any class at a university requires a significant time commitment. Due to the different backgrounds the students have, it is difficult to predict the actual time needed. On average, we see that students spend 6 hours on the class if they do participate every week. Students with little programming experience spend up to 12 hours.

Naturally, you need to balance your course work. If another class has assignments every week and you do not spend at least 6 hours on this class, also you will have a hard time with this class. If you spend less than 6 hours, please, work ahead and do not just spend the 6 hours on another class.

3.3.3 COURSE MATERIAL LIST

Course material is distributed as ePubs, PDF, Video, or Presentations. However, you are required to research some topics on the internet as Cloud Computing is a rapidly evolving field, and you likely find the most up to date information on the Web. However, not all information on the Web is accurate or works. It will be in your responsibility to filter that information.

Throughout the semester we also ask you to help us updating the ePub and to use additional resources as appropriate.

The list of resources for the class include:

- [Cloud Computing](#)[1]
- Introduction to Python [2]
- Linux for Cloud Computing [3]
- Scientific Writing with Markdown [4]

Other documents may include:

- Cloud Technologies, Gregor von Laszewski [[5](#)]
- Project Report Format, Gregor von Laszewski [[6](#)]

3.3.4 HELP

If you take our class, please use Piazza to ask for help. This is important as different Teaching Assistants may answer questions (TAs) based on expertise. Please, do not send e-mail to the instructors. TAs are **not allowed** to answer e-mail send to them personally. On Piazza always use the send to *instructors* group.

3.3.5 HOW TO TAKE THIS CLASS

This class is attended by students with greatly different backgrounds and time schedules. To be most flexible and address all students there are two different ways on how you can take this class.

- Way 1: *Chronological order*. The lecture notes are ordered chronological. Thus you can follow our lecture also in chronological order.
- Way 2: *Free form*. Here you simply look at the Syllabus table for the semester and identity, whatever section you feel like reading (when it becomes available). However, make sure you conduct our weekly **Lab activities**.

Please note that we have set aside regular Lab activities. The activities are pass-fail and will be integrated into your grade. You are certainly allowed to work ahead, but please be aware that based on feedback and observation, we may make modifications to the Labs.

Typically, Lab activities are supposed to be completed within one week as it alerts us of problems, you might have that we can then address.

Lab activities will not receive any credit if you are a residential student, and the activity has not been completed within one week.

However, residential students will get two **Delay a Lab for One Week** passes that you can apply to any of the Labs and still get credit.

If you are an online student, we recommend that you finish the Labs also in one week. However, you will get eight **Delay a Lab for One Week** passes that you can apply to any of the Labs.

Please note that if you would need to postpone a lab for two weeks, you need to use two passes. Lab passes expire one month before the last day of class. You will have to complete all the labs at that time. No credit will be given at that time if this deadline is missed for any delayed Labs as TAs must focus their attention on project support.

Lab passes do not apply to other assignments and due dates.

3.3.6 ASSIGNMENTS

Besides the Lab's, we have only a few assignments in this class. The Lab's will prepare you towards achieving these assignments.

3.3.6.1 Technology Review

Students doing projects related to Cloudmesh are exempt from writing a technology review, but are expected to do more programming and making sure the project has a manual and proper documentation.

As part of cloud engineering, you will be exposed to a large set of technologies. To sharpen your skills in analyzing and evaluating these technologies, you will be asked to prepare a to be determined number of technology reviews that are being added to a class proceeding.

This includes writing a substantial non-plagiarized document that can be added as a chapter to the lecture notes. The review must be done on a topic that is not yet included in our book. The review will not include advertisement statements from those that have developed the technology but will qualitatively describe the technology and potentially contrast it to other related technologies. In addition, you will have to develop an example showcasing how to use the technology. The minimal length of a review is about 800 words. Others will try out your review and test if the example works. Your grade will be influenced by the review of other students. You will also get a grade for the review you provide to other students.

An example of such a section is the

- GraphQL

in the Cloud Computing book.

Alternatively, you can prepare several different smaller sections (at least 5) that may not have an example in it but are more of descriptive nature. Sample sections contributed by students include:

- Section Microsoft Nafik Data Center in the Datacenter Chapter
- Section Lambda Expressions in *Introduction to Python*



It is expected from you that you self identify a section yourself, as this shows competence in the area of cloud computing. If, however, you do not know what to select, you must attend an online hour with us in which we identify a topic with you. Technologies that are not repeatable due to enormous cost or licensing issues need to get prior approval.

This year's technology reviews will center around the use of AI services from major cloud providers.

3.3.6.2 Project

The objective of the project is to define a clear problem statement and create a framework to address that problem as it relates to multi-cloud computing and engineering.

A project is the primary activity that you chose as part of your class. This includes a *project report* or *manual* and working *documented* project code. You will create a significant non-trivial project related to cloud computing and cloud engineering. Up to three students can collaborate. The project could be built on top of a previous project but must have significant additions or modifications. If a previous project is used, a detailed discussion is to be held on what has been improved and is different.

In this class, it is especially important to address the reproducibility of the deployment. A test and benchmark on multiple clouds, possibly including a downloadable* dataset, must be used to verify the correctness of your approach.

The project must demonstrate

1. The use of multiple clouds or cloud services.
2. The implementation of an abstract interface to these clouds or services.
3. The implementation of an OpenAPI Specification and interface.
4. The implementation of an easy deployment.
5. The implementation of a benchmark.
6. The easy replication of the project from source from your short documentation (via the AI's).
7. The integration of the project into Cloudmesh commands. This is super easy.
8. Assume your project has N members then you need to use and provide a testing infrastructure that works on N+1 clouds.

3.3.6.2.1 License

All projects are developed under an open-source license, such as the Apache 2.0 License. You are required to add a LICENCE.txt file and describe how other software if used, can be reused in your project. If your

project uses different licenses, please add a README.md file that describes which packages are used and what licenses these packages have.

3.3.6.2.2 Project Report

For certain projects, the requirement of a report can be waved or is significantly reduced while replacing it with more programming activities. This includes

- *Any project that enhances Cloudmesh*
- *Building a large cloud cluster with Raspberry Pi's (while contributing a manual or a chapter about it)*
- *Any Application project showcasing NIST big data reference architecture use.*

However you still have to do a manual and usage examples, benchmarks and `pytest`s for them.

For all others the following guideline exists:

A project report is to be delivered and continuously improved throughout the semester in GitHub. It includes not just the analysis of a topic, but a short description of the Architecture and code, with **benchmarks** and demonstrated use. Obviously, it is longer than a term paper and includes descriptions about the reproducibility of the application. A README.md is provided that describes how others can reproduce your project and run it. Remember that tables and figures do not count towards the paper length. The following minimal length is required:

- 800 words, one student in the project
- 1200 words, two students in the project
- 1400 words, three students in the project

Projects with more students are expected to do more programming. The report is written in markdown and checked into GitHub. The report will be made available in a class proceeding. A report could be substituted by a

manual and benchmarks upon approval. In this case, a one-page extended abstract has to be written, which includes the link to the manual.

3.3.6.2.3 Project Code

You are expected to deliver a **documented** and **reproducible** code with unit tests that allows a TA to replicate the project with ease. In case you use VM or container images, they must be created from **scratch locally** and may not be uploaded to services such as DockerHub. You can, however, reuse approved vendor uploaded images such as from ubuntu or centos or other Linux distributions. All code, scripts, and documentation must be uploaded to github.com under the class-specific GitHub directory.

3.3.6.2.4 Project Data

Data is to be hosted on IU's Google drive if needed. If you have larger data, it should be downloaded from the internet. It is your responsibility to develop a download program. The data **must** not be stored in GitHub. You are expected to write a python program that downloads the data either from the Web or IU's data storage.

3.3.6.2.5 Work Breakdown

Only needed when you have 2 or more members in your team.

This is a separate document called `breakdown.md` that is included in your project directory. It describes in a bullet list who did what in the project. If you are a team of one, such a document is not needed. It does not count towards the page length of the document. It must include explicit URLs to the git history that documents the statistics to demonstrate more than one student has worked on the project. If you can not provide such a statistic or all check-ins have been made by a single student, the project has shown that they have not adequately used git. Thus, points will be deducted from the project. Furthermore, if we detect that a student has not contributed to a project, we may invite the student (alone) to an oral presentation of all aspects of the project, including a live demonstration of the examples in

real-time to determine the project grade. So if you work in a team contribute and do not have your team members do all your work.

3.3.6.2.6 Bibliography

All bibliography has to be provided in a BibTeX file that **must** either be validated with **jabref** or with **emacs**. Please be advised doing references correctly takes some time, so you want to do this early and throughout the semester. What would take less than 5 minutes a week could quickly add up to multiple hours at the end of the semester. Please note that exports of Endnote or other bibliography management tools do not lead to properly formatted BibTeX files, despite their claims of doing so. You will have to clean them up, and we recommend to do it the other way around. Hence, the easiest way to manage your bibliography is with **jabref*** or *emacs*. Make sure **labels** only include characters from [a-zA-Z0-9-]. Use dashes and not underscore and colons (_ ,:) in the label. Your labels must be meaningful and unique. We will deduct points if you submit an invalid *BibTeX* file to GitHub. So please make sure your file is validated. You can even create your own checks with tools such as `biber`.

We will teach you what to do. It is easy.

3.3.6.2.7 Reproducibility

In general, any project must be deployable by the TAs. If it takes hours to deploy your project, please talk to us before the final submission. This should not be the case. Also, if it takes 100 steps to do by hand, we are sure you can automate them, as you are likely doing something wrong or have not thought about the engineering aspect of cloud computing where as we tend to automate most of the steps. A part of your grade will be how much your project is automated.

The best way to assure reproducibility is to use `pytest`. We will discuss how to do that in class.

We will teach you what to do. It is easy.

3.3.6.2.8 List of Deliverables

You have plenty of time to execute a wonderful project, but you need to work consistently on it. Starting one week before the deadline will be disastrous.

In general, your deliverables will include the following (We will address and explain them in a Lab):

- Provide benchmarks.
- Take results in a cloud services and your local PC (ex: Chameleon Cloud, echo Kubernetes). Make sure your system can be created and deployed based on your documentation.
- Each team member must provide a benchmark on their computer and a cloud IaaS, where the cloud is different from each team member.
- We require you to write one or more pytest's that deploys, run, kill, view, clean that deploys your environment, runs the application, kills it, views the result, and cleans up afterward.
- For python use a requirements.txt file and develop a `setup.py` so your code can be installed with `pip install .`
- For docker use a Dockerfile

We will teach you what to do ...

3.3.6.2.9 Example Outline of a Report

- (If not exempt) write a report that typically includes the following sections:
 - Abstract
 - Introduction
 - Design
 - Architecture
 - Implementation
 - Technologies Used
 - Results
 - Deployment Benchmarks

- Application Benchmarks
 - (Limitations)
 - Conclusion
 - (Work Breakdown)
- Your report will **not** have a *Future Work* section as this implies that you will do work in the future, and your paper is incomplete. Hence we would not grade it. Instead, you can use an optional *Limitations* section.
- Do communicate your status and add a *Workbreakdown* section in which you outline which tasks need to be done and by whom in case of a group project. Once you have done a task simply include maker a task as follows

* [done, Gregor] This was Gregor's task to showcase how to mark it

In case you have an exemption for the project report, you need to use `sphinx` and document your code as part of a manual. We will explain the details in one of our labs.

3.3.7 SUBMISSION

All submissions are conducted via GitHub if not otherwise instructed. Technology reviews are to be added to the `book` GitHub repo with the help of pull requests. The TA's will work with you to integrate them.

As we are working continuously throughout the semester, you must indicate your activities in a README.yml file in your GitHub repo. The GitHub Repo, we will define for you in the first 2 weeks of the semester.

An example of the README.yml file is shown next

```
---
owner:
  firstname: Gregor
  lastname: von Laszewski
  hid: fal8-523-00
  community: i523
  semester: fal8
chapter:
  - keyword: IoT
```

```

  title: Role of Big Data in IoT
  url: https://github.com/cloudmesh-community/fal8-523-00/blob/master/chapter1/paper.md
  group: fal8-523-00 fal8-523-01
- keyword: Datacenter
  title: Green IT data centers in the US
  url: https://github.com/cloudmesh-community/fal8-523-00/blob/master/chapter2/paper.md
  group: fal8-523-00
project:
- keyword: Cloud Cluster
  title: Raspberry PI Cloud Cluster
  url: https://github.com/cloudmesh-community/fal8-523-00/tree/master/project-report
  group: fal8-523-00 fal8-523-01
  code: TBD

```

You **MUST** run `yamllint` on the `README.yml` file. YAML errors will cause point deductions. Any invalid yaml file will result in point deductions. Please keep your yaml file valid at any time. Our scripts depend on it. The yaml file will also be used to create a list for TAs to review your deliverables. If it is not in the YAML file it will not be reviewed. Please note that it is not sufficient to just run `yamllint`, but to compare your yaml file carefully with the `README.yml` examples. Make sure you do the indentation with 2 spaces, do not use the TAB character and make sure you use the list and attribute organization with proper dash placement. Work with the TAs if you have difficulties. If you copy, only copy from the raw content in GitHub. If you work on more

We will teach you what to do. It is easy.

3.3.8 BONUS PROJECTS

This class will not have any bonus projects, as all additional activities should be put in your project or chapter/review contribution. However, we will recognize extraordinary efforts in these activities.

3.3.9 PARTICIPATION

In addition to these artifacts, there will also be a participation component in class that will be determined based on your productive contributions to Piazza to help others that have questions and contributions to the books too, for example, improve sections with spelling, grammar or content. We can see from the GitHub history if you conducted such improvements. Make sure that technical contributions, work on all OSes and are not just targeting a single OS if the improvement is of general nature (exceptions apply).

3.3.10 GITHUB AND PYCHARM MARKDOWN


Please be aware that the ePubs use markdown as defined in pandoc. Neither GitHub nor Pycharm support this version of Markdown. Instead, they use some enhancements that are not supported. Thus do not assume that your GitHub will render in our proceedings. We use the most primitive version of markdown, but use the enhancement for citations and references in citeproc and citeref. We have a docker image that allows you to run the version of markdown on your reports and documents.

The TA's will provide you with support on how to do this. It is trivial.

4 WEEKLY AGENDA

4.1 WEEK 1: COURSE INTRODUCTION

We provide in this document all the activities we do in each week.

Sections marked with a  are not yet released and are likely to be modified.

4.1.1 ASSIGNMENTS

All graded class assignments are posted at

- <https://iu.instructure.com/courses/1858708/assignments>

4.1.2 CLASS VIDEOS

Please watch the following video

-  [Overview Engineering Cloud Computing \(21:05\)](#)

Next, review the following chapters in the book

[Cloud Computing, Gregor von Laszewski](#) [1]:

- Chapter: Preface, ePub Readers
- Chapter: Overview
- Chapter: Definition of Cloud Computing (and its videos)

4.1.3 LAB ACTIVITIES

Summary:

- [] Create an account on <https://github.com>

- [] Fill out the [Class Survey](#)
- [] Post your professional bio to Piazza
- [] Setup your computer
 - [] Install python on your computer (Python 3.8)
 - [] Update pip
 - Install an ePub Reader
 - macOS: Build-in use Books
 - Windows 10: [Calibre](#)
 - others: [Calibre](#)

If you have questions, ask on [Piazza](#) Next, we provide some more details of each of the assignments in this summary.

4.1.3.1 Account Creation

As part of the class, you will need a number of accounts. This includes:

- piazza.com (used for communication)
- github.com (used for the project and other class artifacts)
- chameleoncloud.org (free cloud account)
<https://www.chameleoncloud.org/user/register/>

Optional accounts (apply once you need them, some are time limited):

- google.com (optional)
- aws.com (optional)
- azure.com (optional)
- Watson from IBM (optional)
- google Iaas (optional)

We will never ask you for your passwords.

4.1.3.2 Survey

In this activity, you will be filling out a form with information about you so we can assess how to integrate you in this class best. It is not important that you know any of the technologies we ask you about.

- [Class Survey](#)

4.1.3.3 Bio

This activity serves two purposes. First, it tells us we can communicate with you within [Piazza](#), and second, you can introduce yourself to others in Piazza to potentially build project or study teams.

4.1.3.4 Python 3.8



Please set up a computer on which you can do Python development. We recommend that you use python from <http://python.org>. We will not provide any support for anaconda, as anaconda has hundreds of libraries that we are not interested in using. Also note that conda modifies your environment without telling you. Certainly, you can use virtualbox, or containers in case you like to isolate your development environment from your other systems. Please remember that conda has significant disadvantages while often working with outdated libraries. As the developer of future software, you certainly want to avoid this. If you use python from python.org we require you that you use venv.

The manual may have references to older versions of python. Also it could be that some services do not work in newer python versions yet. It will be your responsibility to find that out and makes sure the services you use have an appropriate python version. We will not accept any project targeting Python 2 or smaller than Python 3.7.4. You must use pip version 19.3.1 or newer. As it may be that newer versions of Python come out during the semester, we like that you install the newest version. At time this note was written this is 3.8.1. SO, if a newer version comes out, please install this.

It may be that some cloud services may not work with that version. In that case it is your responsibility to use or install in parallel a version of python

that works.

4.1.4 ONLINE MEETING RECORDING

-  [Online Meeting Recording for Week 1 2020 \(1:29:39\)](#)
-  [Accepting the GitHub invitation if you can not find it in e-mail, and creation of GitHub issues ToDo lists \(1:29:39\)](#)

Recordings from past years

-  [Online Meeting Recording for Week 1 2019 \(1:29:39\)](#)

4.2 WEEK 2: CLOUD DATA CENTERS

4.2.1 LECTURE MATERIAL

A new version of the following books have been released:

- [e516 Lecture Notes Engineering Cloud Computing, Gregor von Laszewski \[7\]](#):
- [Cloud Computing, Gregor von Laszewski \[1\]](#):

Please read the section in [Cloud Computing\[1\]](#): about **Data Centers**. We decided not to do a video as the material in this section frequently changes, and videos became too quickly outdated.

Do the following assignments:

Due: two weeks, share your findings on Piazza and improve based on the discussions and other students' contributions. We recommend you start in the first week. Gregor will then integrate the contributions into a document that we can then use in the second week for improvements. Please also note that this assignment is used for you to learn markdown with references,

links, and BibTeX entries. Note that GitHub does not support this version of markdown.

- E.Datacenter.2.a
- E.Datacenter.2.b
- E.Datacenter.3
- E.Datacenter.4
- E.Datacenter.5
- E.Datacenter.8

Optional:

- E.Datacenter.9

Optional 

We may post on Piazza examples from last semester that you can leverage and improve. Remember, they may already be outdated by now and could need updates. It will take us some time to publish them. But you can find them already yourself if you look in previous students' repositories.

4.2.2 LAB ACTIVITIES

4.2.2.1 Receive your HID

Due: one week (needs survey to be filled out)

Make sure you have received an HID on GitHub. Look it up at

- <https://github.com/cloudmesh-community>

Make sure to accept the GitHub invitation and try to add a file. You can use either the GUI way with *Create new File* or if you are familiar with GitHub use the command line tools. Next week we will introduce you to convenient tools so you can develop your programs more easily in GitHub.

4.2.2.2 README.yml

Due: one week

Make sure the information in your README.yml file is accurate. Make sure to change the value in the community attribute and use your class number. This will be either e516 or b649 or similar. Please note when we ask you for your hid number, it is the entire hid number, not just the last three digits.

```
---
owner:
  firstname: Gregor
  lastname: von Laszewski
  hid: hid-000
  community: e516
  semester: sp20
```

4.2.2.3 Chameleon Cloud

Due: Now

Make sure you have completed the application for the chameleon cloud as we will use that in our lab soon.

4.2.2.4 Review Python

Due: throughout the semester

This activity will go on throughout the entire semester. You do not have to be a Python expert within one day, but if you have knowledge gaps, please review and consult our Python Book. Start with the language features.

- [Python for Cloud Computing, Gregor von Laszewski \[2\]](#):

We typically use Python 3.8; the book may be a bit outdated in that aspect.

4.2.2.5 Setup your Computer

4.2.2.5.1 Python

Due: One Week

Make sure you have Python set up. Read up on virtualized python environments as used in Python 3.8. Its super simple. Understand what the command

```
$ cd ~  
$ python -m venv ENV3  
$ source ENV3/bin/activate
```

Those using Windows, please find out how to do it on Windows, provide a document `venv-windows.md` in your HID directory describing it.

Those using conda/anaconda, please find out how to do it with conda/anaconda, provide a document `venv-conda.md` in your HID directory describing it.

Make sure you create a separate venv for this class called ENV3 that we will use for this class and should not be used for other classes as we do not want to create side effects. Please also be aware that at times it may be necessary to delete your python environment in case you do something wrong and it would be unwise to combine all your python activities into one python install.

As you can see, this is super easy.

We recommend PyCharm for an IDE if you have not yet picked one. PyCharm automated code improvements will be used for most of our projects. It really helps you! We will teach you throughout the semester how to use it.

4.2.2.6 Ubuntu Multipass

Due: one week

Locate the Section about Multipass in the book

- [Cloud Computing, Gregor von Laszewski \[1\]](#):

and read it.

Install ubuntu multipass on your computer. Note that you must have at least 8GB of main memory on your computer. However, you may need to close chrome or PyCharm. Please monitor your consumption of memory to determine if you like to buy more memory for your computer and upgrade it. A computer with 4GB is likely not to be usable for this assignment.

In case you do not have a computer on which you can execute this, please consult with us in Piazza. You may have to do more work than other students.

For example (a) you may have to immediately start using chameleon cloud and learn OpenStack as an alternative; (b) you may need to use a virtual box; (c) or you may have to buy yourself a cheap up to date computer on which you can execute the class. You must be able to be superuser/administrator of that computer. The computer must support virtual machines and Docker.

Please also note that if you have Windows Home 10 on your system, you need to upgrade it to Pro or EDU. Make sure to configure it as 64 bit.

It is in your responsibility to have the proper equipment.

Analogy: When you play tennis, you get yourself a tennis racket, and you do not show up with a pingpong paddle to play it.

Do Assignments:

- E.Multipass.1
- E.Multipass.5

4.2.2.7 Optional: Plagiarism Certificate

Due: optional, but if you do not understand what plagiarism is, please find out.



At IU you are required to know these concepts before you take any class. Thus the excuse: “I did not know that this is plagiarism” does not apply to any class at IU including this one.

When writing contributions that can be integrated into the class material, it is important that it is not plagiarized. It is most important that you not just copy content from Web pages, but make appropriate modifications and provide credit to where you found this information.

As you certainly will have learned from other classes what plagiarism is and how not to plagiarize, you will probably be fine. However, we often find one or two students in a class that does not know what it is. Therefore we **STRONGLY RECOMMEND** that you take the plagiarism certificate offered by IU. This has the advantage that you can show it in other classes and show you are informed. Please also read the code of conduct rules at IU.

For this reason, we included this material also in part in the Book


- [Scientific Writing with Markdown](#)

Please read the chapter about Plagiarism.

Please be aware that we may conduct superior plagiarism tests. Our tests are even better than *Turn-it-in* while we are able to identify *translations* from other languages, copies of text from non-public external university servers, and company archives that turn-it-in does not check. IU has a strict policy that we must follow. Plagiarism/cheating could lead to expulsion from the university. The argument *I did not know what plagiarism is* does not count according to IU rules. You must know what it is prior to you taking a course at IU.

It is not the subject of this class to teach you what plagiarism is. We do this just to remind you to avoid any uncertainty about it.

4.2.3 ONLINE MEETING RECORDING

 this is a recording from last semester. A new recording will be published here for this semester once it becomes available.

The online meeting recording from Tue Sep 3, 2019 is available from this link:

-  [Online Meeting Recording for Week 2 \(48:38\)](#).

This meeting lasted actually half an hour longer but included duplicated questions that we removed.

4.3 WEEK 3: CLOUD COMPUTING ARCHITECTURES



Goals:

- Start of Chapter Selection
- Start of Project Selection
- Review Python

4.3.1 LECTURE MATERIAL

A new version of the following books have been released:

- [e516 Lecture Notes Engineering Cloud Computing, Gregor von Laszewski](#) [7]
- [Cloud Computing, Gregor von Laszewski](#) [1]
- [Introduction to Python for Cloud Computing, Gregor von Laszewski](#) [2]

Reading Assignments:

1. Read in the book [Cloud Computing](#) [1] the chapter about Cloud Architectures.
2. Read in the book [Introduction to Python for Cloud Computing](#) [2] the chapter about the Python language
3. Focus your on refreshing your python knowledge about lists, set, dict, and classes.

4.3.2 LAB ACTIVITIES

You should in the first week of the class set up python 3.8 on your computer. If you have not yet done so, please do. We recommend that you use python from python.org. If you use conda or anaconda it is up to you to figure out how to do this. You can post guides to that in piazza and we will include them possibly in future lecture notes.

Whatever you do you **must** use a python virtual venv.

4.3.2.1 Review: venv in python 3 (Graded)

To create a venv please use the command

```
$ python -m venv ~/ENV3
```

Please understand the following concepts (no submission needed):

- How do you activate the virtual env in your OS?
- How do you modify your `.bashrc` file so that the python venv is loaded automatically upon start of a new terminal?
- In case you use zsh either switch to bash or describe how do you modify zsh so that the python venv is loaded.
- Why do you need to use `venv` for this class? Provide a one paragraph answer in your notebook.md file.

4.3.2.2 Review: Anaconda (Graded)

This lab only has to be done by those using anaconda/conda. Those using python from python.org do not have to do this assignment.

Please provide your answers in the notebook.md file.

- What problems may you encounter when using anaconda as python developer?
- Let us assume you use anaconda on your virtual machines in the cloud. Let us assume you start 1000 vms all using anaconda. What is the overhead in wasted space on these machines if you just wanted to use as simple regular python program on these VMs?
- How do you find out how much space is used by your program and its libraries?
- How do you switch between anaconda and regular python 3.7.4
- What is the difference between conda, miniconda, anaconda?
- Why do you want to use a virtual environment even for conda/anaconda?

4.3.2.2.1 Dicts

We like to remind you about dicts in python.

Read up on dicts and experiment with them.

- How do you merge the content of two dicts?

4.3.2.2.2 f-Strings in Python 3

Python 3 provides some very nice way of using variable names as part of string manipulations.

- Locate the PEP 498 and study it:
<https://docs.python.org/3/whatsnew/3.6.html#whatsnew36-pep498>

Try out the following

```
test = "Gregor"
msg = f"This is a test {test}"
print (msg)
```

```
def f(test):
    msg = f"This is a test {test}".format(**locals())
    print (msg)
```

```
from cloudmesh.common.debug import VERBOSE
d = {"test": "Gregor"}
VERBOSE(d)
```

In one of the examples `locals()` is used.

- What does `locals` do?
- What does `**` in the format statement do?

4.3.2.2.3 Classes

This can be completed at a later time throughout the class

- What is `self` in classes?
- Why does `self` needs to be used in regular method definitions in classes?
- What can I do with **init** and why is it used?
- What is `cls` and `@classmethod`?
- Why would one use `@staticmethod`?

4.3.2.2.4 Python Modules

This can be completed at a later time throughout the class

- What is a `setup.py` file
- What is the difference between `pip install .` and `pip install -e .`
- How do I uninstall a python module?
- My python virtual environment is broken, What do you do now?

We will be discussing these question in the Labs (online/and residential).

4.3.2.3 Graded Activities

In the book [Introduction to Python for Cloud Computing \[2\]](#) please do the following assignments

- E.Cloudmesh.Common.1
- E.Cloudmesh.Common.2
- E.Cloudmesh.Common.3
- E.Cloudmesh.Common.4
- E.Cloudmesh.Common.5
- E.Cloudmesh.Shell.1
- E.Cloudmesh.Shell.2
- E.Cloudmesh.Shell.3

4.3.3 CHAPTER SELECTION

As part of this class you will be conducting a technology review while writing a chapter that looks at a topic related to cloud computing. The chapter is written in markdown and placed in your HID directory at

```
chapter/report.md
```

please note that all charaters of the filename are lower case.

This years topic focusses mostly around *Cloud AI Services* but we also accept other topics that you may find interesting and are not covered in substantial length in any of our books.

We have posted here some initial ideas for chapter contributions. Remember that you **must** not write an introductory paragraph to your contributions as they are most likely already covered in the books. Instead, you **must only** focus on the topic at hand.

So please visit the link and select a topic or suggest your own

<https://docs.google.com/spreadsheets/d/1QxIFCSQI66-zR9H6uI9-N5F99fi23o3enrU51QJdQ7c/edit?usp=sharing>

Please only fill out the white lines and leave the yellow once untouched. If there are more than two people in a category, please coordinate with each other how to write it while only providing a single document. Depending on what the topic is about, you may have to expand it. Please note that this is a multipage activity and not a one-sentence activity. References must be included as proper references at the end, we will teach you in time on how to do that, Examples are given in the markdown book.

All chapters need to include

- a writeup that introduces the technology without plagiarism and advertisement claims by the developers of that technology.
- an example on how to use it that can be replicated and you have tried yourself if possible.
- references in bibtex format

Due dates:

- Feb 3rd: Last date to propose a chapter
- Feb 10th: Last date to submit the first draft
- Feb 17: Last date to return a peer review (you get a grade from the author of the chapter, the reviewer will also check your examples)
- Feb 24: Last date to hand in the final version of the chapter

4.3.4 PROJECT SELECTION

Deadline: Feb 7 last day to hand in your project proposal

You will be selecting a cloud related project over the next 2 weeks that you will be developing until the end of the semester. The project must have the following requirements:

- Programming should be done using python, if another programming language is used, please contact us and justify the use. You will also

have to do part of your programming to, for example, coordinated the deployment or the benchmark likely in python.

- The project must use OpenAPI 3.0 to define a REST service. We will teach you how to do that in a future activity.
- The project must use connexion to automatically generate the rest service (please do not use swagger codegen). We will teach you how to do that in a future activity.
- The project must use at least $1 + n$ clouds for each team member. Where n is the number of team members with maximum 3 team members. We will teach you how to do that in a future activity.
- The code must use pytest. We will teach you how to do that in a future activity.
- The code must use `cms sys generate`. This is one of the lab assignments for this week.
- The code must use the `cloudmesh benchmark/stopwatch` class. This is one of the lab assignments for this week.
- If an AI service is used we recommend to use scikit-learn or Kearos and contrast it with AI services offered by cloud providers. This will be discussed in upcoming lectures.
- You must provide a report with meaningful benchmarks.
- You will have to write a report. We will discuss with you how to do that in a future lecture.

Please note that the above requirements also hold true if you use technologies such as Hadoop, Spark, Kubernetes, AWS, Azure, Google, OpenStack, ...

Certain projects will have custom deliverables that we will refine with you once you have chosen such a project. For example, if you were to chose a cloudmesh project you will be asked to develop a manual instead of a report.

A preliminary list of projects is available at.

- <https://cloudmesh.github.io/cloudmesh-manual/projects/>

We will add additional project ideas once the become available. Please note that we consider the cloudmesh related projects easy as we introduce you gradually in all aspects as part of the class to deliver a successful project.

Those wanting to chose the Raspberry PI Cluster or the Robot Boat, please contact instructors via piazza. We want to set up a meeting in MESH to better discuss this project.

When it comes to the scope of the project, remember a project takes up to 12 weeks to be completed. It is not allowed to just search on `github` or another book and “replicate” a project done by someone else. Your project must have a novel component. Please note this class is called Engineering Cloud Computing we must see clearly an aspect that you engineer the cloud, e.g. the setup of a reproducible cloud environment is mandatory. Please ask questions and understand this. While you are able to use all services, all images must be created from scratch and we must be able to reproduce them. We will decline all projects that point us to images that you ask us to download and uploaded by you on github, dockerhub or similar. Instead you must provide us with scripts, dockerfiles, makefiles or similar that create the images. You are allowed to use images hosted by major vendors such as an ubuntu19.04 image or on chameleon cloud the images starting with cc-* and so on ...

4.3.5 WORKING AHEAD

Obviously we will be introducing you to some more advanced concepts that are not yet finalized in the books. You can certainly use documents form the

internet to learn about such concepts.

Concepts we will need are listed in the Syllabus.

On the python side, we will introduce you to

- pytest
- github API as an example for a REST service
- Azure python API
- AWS boto
- OpenstackSDK API
- Google cloud API
- ...

On the cloudmesh side we will introduce you soon to

- cloudmesh.yaml as preliminary documented in the cloudmesh manual
- cloudmesh cloud bundle
- cloudmesh storage bundle

4.3.6 ONLINE MEETING RECORDING

 This is a recording from last year

This meeting took place Sep 10, 2019, 8-9pm EST

-  [Online Meeting Recording for Week 3 \(1:54:01\)](#)

4.4 WEEK 4: OPENSTACK

4.4.1 VIDEO

In case you have not yet created an ssh key, the following video is useful. This can be replicated on any Linux, macOS and Windows 10 machine, in case of Windows 10 use gitbash

-  [SSH keygen \(4:07\)](#).

A Video briefly summerizing an introduction to cloudmesh for multi cloud environments including pointedts to projects related to virtual directories, compute services, and virtual clusters

-  [Cloudmesh Version 4 \(44:01\)](#).

A video on how to start an login into a virtual machine on Horizon is presented next:

-  [OpenStack Horizon \(10:49\)](#).

A video how to start vms with cloudmesh is available here:

-  [Cloudmesh cms vm boot \(15:07\)](#).

A number of videos on explaining the internals of cloudmesh is available here:


The first video introduces you to cloudmesh common a library to execute commands on your os and interact with it

-  [cloudmesh-common \(13:40\)](#).

The second video explains details about cloudmesh commpute and also tells you about what is expected for the project to be developed. However the video does not yet explain how to do a virtual cluster from the Providers, which is one goal for this semester.

-  [cloudmesh-cloud \(15:09\)](#).

The third video tells you about the cloudmesh-storage project that is supposed to be don in the project, this includes

- better tests
- completion of additional providers
- using of Processpool and mongodb for managing files to be copied
- the creation of a REST service that includes the pool to allow copy from one cloud to another.
-  [cloudmesh-storage \(8:03\)](#)

4.4.2 LECTURE MATERIAL

- A Book for [Chameleon Cloud](#) is available

A new version of the following books have been released:

- [e516 Lecture Notes Engineering Cloud Computing, Gregor von Laszewski](#) [7]
- [Cloud Computing, Gregor von Laszewski](#) [1]
- [Introduction to Python for Cloud Computing, Gregor von Laszewski](#) [2]

4.4.3 LAB: OPENSTACK

This week we will be learning how to manage virtual machines on OpenStack. You are requested to explore the GUI interface which is called horizon so you can. verify your activities easily in case you have issues with the command line tools. However, our main goal will be that you use command line tools to interact with Chameleon cloud.

Whatever you do after you are done with the VMs you need to terminate them so you do not unnecessarily waste compute time.

4.4.4 NAMING OF VMS

Your HID is of the form fa16-516-NNN or similar for other classes. Please note that the number is unique across classes. This identifies you and if you

start a vm in a shared space such as chameleon we can use it to identify people and notify them easily.

Thus please use the following naming scheme

```
NNN-firstname-i
```

where topic is a topic for the vm such as `webserver` and i is a number such as `1` as you may start multiple vms.

Please never start more than 3 vms without consultation with Gregor as we will run out of node hours for the class if we do so.

4.4.5 HORIZON

The information on how to use Horizon is available in the Chameleon book.

A video of the meeting on Tuesday 17 Sep will be made available

4.4.6 CLOUDMESH OPENSTACK INTERFACE


The information on how to use it is available in the Chameleon book.

A video of the meeting on Tuesday 17 Sep will be made available

4.4.7 OPENSTACK COMMAND LINE CLIENT

The information on how to use it is available in the Chameleon book.

4.4.7.1 Installation of Cloudmesh Cloud Bundle

 Do these only after you have completed the cloudmesh shell related assignment from last week.

Compute:

```
$ cd cm
$ cloudmesh-installer git clone cloud
```

```
$ cloudmesh-installer install cloud
```

The next is optional and only for those that chose a cloudmesh storage related project. The installation of the `storage` bundle includes the installation of the `compute` bundle.

```
$ cd cm
$ cloudmesh-installer git clone storage
$ cloudmesh-installer install storage
```

To see the available commands type

```
$ cms help
```

4.4.7.2 SSH

Make sure you have a password protected ssh key

```
$ ssh-keygen
```

4.4.7.3 Configuration

1. Change the username and password for the chameleon cloud in
`~/.cloudmesh/cloudmesh.yaml`
2. Change the password in the mongodb section
3. Change the information in the profile section

4.4.7.4 Cloudmesh Mongo

Setting up cloudmesh Mongo is discussed in the cloudmesh manual. We suggest you do the one discussed in the distribution section for your system. It is actually built into cloudmesh, but before you do it, we suggest you backup your machine.

```
$ cms admin mongo install
```

Once you set it up use `cms init` which wipes the db and should only be used once

```
$ cms init
```

after that always use

```
$ cms start
```

```
$ cms stop
```

4.4.7.5 Start a VM

OpenStack could be over-utilized and that a VM may not start before a timeout. Gregor observed 70% success rate.

Task: whenever you start a vm, please keep a record if it started or not

do this in a file on your GitHub called chameleon-success.md

```
success: 7  
failure: 3
```

make sure the failures are not recorded based on programming errors or wrong parameters.

Use the commands

 switch on debugging and trace

```
cms set cloud=chameleon  
cms image list --refresh  
cms flavor list --refresh  
cms vm boot  
cms image list --refresh
```

 Explain the difference between `--refresh` and not using it.

4.4.8 WORKING WITH THE VM

Log into the vm with

```
cms ssh
```

4.4.9 EXERCISE

1. Start a vm with Horizon, login, and terminate it
2. Start a vm with Cloudmesh, login, and terminate it
3. Use robo3T or a similar program to brows in the Cloudmesh MongoDB

4.5 WEEK 5: CLOUD VM COMPUTE SERVICE

4.5.1 VIDEOS

Watch the videos from week 4 related to Cloudmesh

4.5.2 LECTURE MATERIAL

A new version of the following books have been released:

- [e516 Lecture Notes Engineering Cloud Computing, Gregor von Laszewski](#) [7]
- [Cloud Computing, Gregor von Laszewski](#) [1]

Please read the chapter in the book [Cloud Computing](#) [1] about

- **Hypervisor**
- **IaaS**, pick one cloud you like to get familiare with and focus on that

1. Do not read the sections marked with



Construction

2. Do not use **Python LibCloud** we want you to find the vendors python libraries whcih are typically superior and have better access to the information related to IaaS for the provider. This will make your projects much easier in the long run.

4.5.3 LAB: VM COMPUTE SERVICES

4.5.4 EXERCISE

1. Find a cloud you like and identify the native Python API (but not libcloud, in case of openstack use the openstack sdk which is new and not nova and the other api's).
2. Demonstrate a Python program to list images, flavors and virtual machines. You can create a vm via the GUI on your cloud, name it with your firstname.
3. When it comes to credential management, please use cloudmesh.yaml and Config(). Learn how to do that. Under NO CIRCUMSTANCES post your passwords or add files to GitHub that include your passwords or other credentials.
4. Develop a python program that adds keys and security groups (difficult).
5. Showcase that you can ssh into the vm (difficult)

This exercise can be done openly in class and everyone can share code with everyone, as long as you acknowledge the student with name and hid.

4.6 WEEK 6: REST

4.6.1 LECTURE MATERIAL

A new version of the following books have been released:

- [e516 Lecture Notes Engineering Cloud Computing, Gregor von Laszewski](#) [7]
- [Cloud Computing, Gregor von Laszewski](#) [1]
- [Chameleon Cloud](#)
- [Introduction to Python for Cloud Computing, Gregor von Laszewski](#) [2]

Go through Chapter **REST** of the [Cloud Computing](#) [1] book. This chapter includes,

- An introduction to REST

- An overview of frameworks using REST
- An overview of OpenAPI and how it relates to REST
- An introduction of using GitHub which is also a REST service from a Python API.

The chapter also includes a section on OpenAPI2 and Eve which is however for this class not relevant.

4.6.2 LAB ACTIVITIES

Implement the REST service explained in the Section *OpenAPI REST Service via Introspection* in your local machine

- Complete the exercises:
 - OpenAPI.Conexion.1
 - OpenAPI.Conexion.2
 - OpenAPI.Conexion.3
 - OpenAPI.Conexion.4
 - OpenAPI.Conexion.5
- Complete the exercises:
 - E.OpenAPI.1
 - E.OpenAPI.2
 - E.OpenAPI.3
 - E.OpenAPI.4
 - E.OpenAPI.5

4.6.3 GRADED LAB ACTIVITY

In this lab activity you will be adding more functionality to the REST cpu example and deploying the service in the Chameleon Cloud.

You may need to install [py-cpuinfo](#) library to help you collect information for the service implementation.

1. Change the *cpu* GET method, to work in a operating system invariant way (i.e. use python libraries to determine the CPU name, rather than system calls)
2. Play around with *py-cpuinfo* library in Ubuntu environment just to get an idea about the information you could retrieve. I suggest you spawn an Ubuntu 18.04 container on your local machine for this.
3. Add a GET method to get cache size of the CPU. URL parameter *{level}* should specify the cache level.

GET `http://localhost:8080/cloudmesh/cpu_cache/{level}`

- level = 'l3' for l3 cache size
- level = 'l2' for l2 cache size

The return should be a dictionary as follows.

```
{  
  "l3" : "8448 KB"  
}
```

4. If cache level is not specified, the following dictionary should be returned.

```
{  
  "caches": {  
    "l3": "8448 KB",  
    "l2": "1024 KB"  
  }  
}
```

5. Add these methods to the *cpu.yaml* definition.
6. Deploy the service locally and test the services using *swagger-UI* and *curl*.
7. Create a **m1.small** instance in Chameleon Cloud with **ubuntu 18.04** image. Deploy your new web service in the cloud instance. You should use *cloudmesh* commands to start these VMs.

8. Use curl to call the webservice in the cloud. (You are NOT expected to expose the service through the public IP address)
9. Capture the outputs for each service paths in a meaningful way (images, screenshots, etc) and compile a Markdown file. The Markdown should include, the new cpu.yaml file, server.py, and cpu.py files together with outputs. Furthermore, you should include the *cloudmesh* commands that you have used in the process and you may include improvements to the service such as handling malformed requests, etc.
10. In the Markdown file, discuss what are the ways you could add security for these services (You do NOT need to implement this).

4.7 WEEK 7: MAPREDUCE





4.7.1 LECTURE MATERIAL

A new version of the following books have been released:

- [e516 Lecture Notes Engineering Cloud Computing, Gregor von Laszewski \[7\]](#):
- [Cloud Computing, Gregor von Laszewski \[1\]](#):

Please read the **Hadoop** sections in [Cloud Computing\[1\]](#)

Watch these online videos for Hadoop MapReduce:

-  [Map Reduce, Hadoop, and Spark \(19:02\) Hadoop A](#)
-  [Hadoop 13:19 Hadoop B](#)
-  [Hadoop 12:57 Hadoop C](#)
-  [Hadoop 15:14 Hadoop D](#)

Note: All Map Reduce/Hadoop Lab activities are optional

4.7.2 LAB ACTIVITY: HADOOP INSTALLATION

The following exercises will guide you to install Hadoop on a single node and then run a MapReduce job in Python. Please figure out the required command lines. These commands are available in the book sections: [Cloud Computing, Gregor von Laszewski, Ed. 2019](#) [1]

4.7.2.1 Exercises:

- Install prerequisite software
 - Install Java
 - Install SSH
 - Install Maven
- Configure programming environments
 - `JAVA_HOME`
 - `core-site.xml`
 - `hdfs-site.xml`
 - SSH localhost without a passphrase
- Start Hadoop HDFS
 - Format HDFS filesystem
 - Start NameNode and DataNode via command lines
 - Check NameNode status via Web Interface: NameNode - `http://localhost:9870/`
- Start Hadoop YARN
 - Configure
 - `mapred-site.xml`
 - `yarn-site.xml`
 - Start YARN via command line
 - Check YARN status via Web Interface: ResourceManager - `http://localhost:8088/`

After finishing all these steps, you are good to move forward to MapReduce programming.

4.7.3 LAB ACTIVITY: PYTHON MAPREDUCE

See in the [map Section of the Python book](#):

The basic syntax of the map function expects a function object and any number of iterables like a list or dictionary. It executes the function object for each element in the sequence and returns a list of the elements modified by the function object.

```
def multiply(x):  
    return x * 2  
  
map(multiply, [2, 4, 6, 8])  
  
# Output [4, 8, 12, 16]
```

4.7.4 LAB ACTIVITY MAP/REDUCE ON THE CLOUD

4.7.4.1 Python Word Count in MapReduce

```
#!/usr/bin/env python  
  
import sys  
for line in sys.stdin:  
    line = line.strip()  
    words = line.split()  
    for word in words:  
        print('%s\t%s' % (word, 1))
```

This code is used as Mapper.

```
#!/usr/bin/env python  
  
from operator import itemgetter  
import sys  
current_word = None  
current_count = 0  
word = None  
for line in sys.stdin:  
    line = line.strip()  
    word, count = line.split('\t', 1)  
    try:  
        count = int(count)  
    except ValueError:  
        continue  
    if current_word == word:  
        current_count += count  
    else:  
        if current_word:  
            print('%s\t%s' % (current_word, current_count))  
            current_count = count  
            current_word = word  
        if current_word == word:  
            print('%s\t%s' % (current_word, current_count))
```

This code is used as Reducer.

Please note these code snippets are used to demonstrate the idea of Mapper and Reducer in Python. It leaves some bugs by intentions. Please debug the above code or write your own version of MapReduce.

4.7.4.2 Run Hadoop MapReduce in Python

```
bin/hadoop jar <path_to_hadoop_libs>/hadoop-*streaming*.jar \  
-file /<path_to_mapper>/mapper.py \  
-mapper /<path_to_mapper>/mapper.py \  
-file /<path_to_reducer>/reducer.py \  
-reducer /<path_to_reducer>/reducer.py \  
-input <input_file_path> \  
-output <output_file_path>
```

4.7.5 LAB ACTIVITY: MAPREDUCE ON THE CLOUD

4.7.5.1 Hadoop Cluster Setup

Hadoop's Java configuration is driven by two types of important configuration files:

- Read-only default configuration - core-default.xml, hdfs-default.xml, yarn-default.xml and mapred-default.xml.
- Site-specific configuration - etc/hadoop/core-site.xml, etc/hadoop/hdfs-site.xml, etc/hadoop/yarn-site.xml and etc/hadoop/mapred-site.xml.

Additionally, you can control the Hadoop scripts found in the bin/ directory of the distribution, by setting site-specific values via the etc/hadoop/hadoop-env.sh and etc/hadoop/yarn-env.sh.

To configure the Hadoop cluster you will need to configure the environment in which the Hadoop daemons execute as well as the configuration parameters for the Hadoop daemons.

HDFS daemons are NameNode, SecondaryNameNode, and DataNode. YARN daemons are ResourceManager, NodeManager, and WebAppProxy. If MapReduce is to be used, then the MapReduce Job History Server will also be running. For large installations, these are generally running on separate hosts.

4.7.5.2 Configuration

Configure all Hadoop daemons:

- NameNode: HDFS_NAMENODE_OPTS
- DataNode: HDFS_DATANODE_OPTS
- Secondary NameNode: HDFS_SECONDARYNAMENODE_OPTS
- ResourceManager: YARN_RESOURCEMANAGER_OPTS
- NodeManager: YARN_NODEMANAGER_OPTS
- WebAppProxy: YARN_PROXYSERVER_OPTS
- Map Reduce Job History Server:
MAPRED_HISTORYSERVER_OPTS

Configure Namenode to use parallelGC and a 4GB Java Heap, the following statement should be added in hadoop-env.sh:

```
export HDFS_NAMENODE_OPTS="-XX:+UseParallelGC -Xmx4g"
```

4.7.5.3 Operating the Hadoop Cluster

To start a Hadoop cluster you will need to start both the HDFS and YARN cluster. The first time you bring up HDFS, it must be formatted. Format a new distributed filesystem as hdfs.

- Format NameNode (on a dedicate node in the cluster)
- Start NameNode (on a dedicate node in the cluster)
- Start DataNode (on each node of the cluster)
- Start ResourceManager (on a dedicate node in the cluster)
- Start NodeManager (on each node of the cluster)

4.7.5.4 Run Hadoop MapReduce in Python

```
$ bin/hadoop jar <path_to_hadoop_libs>/hadoop-streaming.jar \  
-file /<path_to_mapper>/mapper.py \  
-mapper /<path_to_mapper>/mapper.py \  
-file /<path_to_reducer>/reducer.py \  
-reducer /<path_to_reducer>/reducer.py \  
-input <input_file_path> \  
-output <output_file_path>
```

4.8 WEEK 8: PROJECT REVIEW

Make sure to have made major progress in your project

Document it in

- github by updating report.bib
- github by checking in your current code

4.9 WEEK 9: CONTAINERS - DOCKER

4.9.1 LECTURE MATERIAL

A new version of the following books have been released:

- [e516 Lecture Notes Engineering Cloud Computing, Gregor von Laszewski \[7\]](#):
- [Cloud Computing, Gregor von Laszewski \[1\]](#):

Please read the section in [Cloud Computing\[1\]](#): about **Containers**. We decided not to do a video as the material in this section frequently changes and videos became too quickly outdated.

4.9.2 LAB ACTIVITY DOCKER

Do the following assignments with all its questions:

- E.Docker.1
- E.Docker.2
- E.Docker.3

Place your solutions in your github hid directory

4.9.3 LAB ACTIVITY CONTAINER REST SERVICE

Develop a simple REST service using OpenAPI. Use the “Introspection” principal with the connexion web service. Develop a container that runs the REST service. Write a script or pytest that contacts the container and returns the result from the REST service.

Write a Dockerfile that create the service

Write a cloudmesh command (remember we did this before) that manages and interacts with the server, but it actually interacts with the container and not just the native service.

Here is an example list sof commands (replace myservice with a comand that is not used already in cloudmesh-cloud or cloudmesh-storage:

```
cms myservice start
    starts the service

cms myservice stop
    stops the service

cms myservice PATH
    Connects to the running service and returns the Object

curl ...
    Document how you interact with your service
```

Obviously you can use services from your project. A more elaborate version of tis will be in your final project. Develop commands that make sense for you, This can include get, upload, delete, and update actions for example.

4.10 WEEK 11: GRAPHQL AND MESSAGING

4.10.1 LECTURE MATERIAL

A new version of the following books have been released:

- [e516 Lecture Notes Engineering Cloud Computing, Gregor von Laszewski \[7\]](#)
- [Cloud Computing, Gregor von Laszewski \[1\]](#)

Please read the chapter in the book [Cloud Computing \[1\]](#) about GraphQL and Messaging.

We have spend significant time to teach you how to generate REST services from OpenAPI specifications. This has the advantage that you do not have to develop a server as it can be outomatically generated from the OpenAPI specification.

Furthermore clients could be developed with Swagger codeGen in the supported large number of supported programming languages. However using Swagger codeGen is not required for this class and goes beyond the scope we like you to explore as part of this class. So if you are using Swagger code gen, you need to contact Gregor to identify why you would need this for this class. In general it is not needed.

Instead we like to focus on two different technologies. One is called GraphQL and the other is called MQTT.

The information about them are included in the book:

- [Cloud Computing, Gregor von Laszewski \[1\]](#)

Please locate the chapters and read

4.10.2 LAB ACTIVITY GRAPHQL

Before doing these optional assignments, make sure that your project is in very good shape. These optional assignments are ungraded.

- E.GraphQL.1
- E.GraphQL.3

4.10.3 OPTIONAL LAB ACTIVITY MQTT

Before doing these optional assignments, make sure that your project is in very good shape. These optional assignments are ungraded.

- E.MQTT.2

4.11 WEEK 12: GO

As you have noticed cloud computing has a lot to do with parallel computing, servers, and messaging.

While python is a great language for cloud computing we also like to make you aware of some other languages. One of these languages is GO. Go has a lot of features included from CSP.

Please read the chapter in the book [Cloud Computing \[1\]](#) about Go.

4.11.1 OPTIONAL LAB ACTIVITY GO REST SERVICE

Before doing these optional assignments, make sure that your project is in very good shape. These optional assignments are ungraded.

- E.Go.1

4.12 WEEK 13: FAAS

Recently you have heard about Function as a Service. We will provide some introductory material and discuss some concrete FaaS frameworks and FaaS offerings by cloud providers.

Please read the chapter in the book [Cloud Computing \[1\]](#) about FaaS.

4.12.1 LAB ACTIVITY

The rest of the semester you can explore aspects we taught you and focus on your projects.

4.13 WEEK N - 16

The agenda for the next weeks will likely follow the Syllabus. Adjustments may need to be made based on pace of the class.

We will try to release new material on Tuesdays's before midnight. Assignments will than be due on Tuesday the follwoing week at 9am in the morning. If things are unclear please ask questions on Piazza.

5 APPENDIX

5.1 FAQs

5.1.1 WHERE DO I FIND THE MATERIAL FOR THIS CLASS?

- <https://piazza.com/class/jzKFveoqwri3e4>

5.1.2 WHERE TO FIND THE WEEKLY AGENDA?

We have lecture notes for this class distributed as epub and PDF. We recommend creating for the link shortcuts in your Web browser so you can click on them quickly. However, all resources are posted at

- <https://piazza.com/iu/spring2020/e516spring2020/resources>

You can find all links and more also at

- <https://laszewski.github.io>

5.1.3 WHEN DO THE ONLINE MEETINGS TAKE PLACE?

TBD: see: <https://piazza.com/iu/spring2020/e516spring2020/resources>

Online meetings are for residential students not a substitute for the Friday Meetings.

5.1.4 WHERE CAN I FIND THE RECORDINGS OF THE ONLINE MEETINGS?

The meeting recordings are generally available two days after the meeting. However, if the books include the information, we may opt not to publish them and instead refer to the book.

The recordings are added to the Lecture notes.

5.1.5 Do I NEED TO READ THE CLASS MATERIAL TO TAKE THE CLASS?

Yes, this is a real question from a past student. IU does not offer classes that give you a certificate without understanding the material. To understand it, you must start by reading it.

5.1.6 LAB: WHY CAN I NOT SOLVE AN ASSIGNMENT IN 2 MINUTES?

In some cases, it takes you a **considerable** effort to research the answer. An example is the Datacenter question about energy consumption or cost. It is not expected that google may return a result quickly. You may need to consider other publications such as google scholar, or professional journals. If you, after some hours, have not found an answer, maybe there exists no such answer. Check what other students have done and ask them how they found the information. Remember, this class is a community. Discuss on Piazza.

5.1.7 LAB: HOW MUCH TIME DO I NEED TO SPEND ON EACH QUESTION?

This question can not be answered as it depends on the question. Some questions may take hours. The time students typically spend on this class is 6-8 hours. However, sometimes it may be less or more dependent on your background. Please be reminded that many students with many different backgrounds attend this class. For example, if we ask you to do some python programming, and you have no background in Python, that task may take you longer than someone who has Python background. However, after you have done some Python tasks, the next python task will be easier. Let us assume you do not know Python, but it took you only 1 hour to do the assignments for the week, we recommend that you do the rest of the time to learn Python. This way , when the python assignment starts you are a Python expert also.

5.1.8 I DO NOT UNDERSTAND PIAZZA?

Attend the online hours and ask questions there or watch the recorded online meetings as we explained it there also.

5.1.9 WHY YOU SHOULD NOT READ THE PIAZZA MAIL BUT USE CLICK HERE INSTEAD?

Piazza sends you e-mails. You can control the frequency in the piazza settings. However, sometimes we do make corrections to the post after you received the e-mail. To prevent any issues, we do recommend that instead of reading the e-mail, you just use the click here feature, which is placed on top of each mail.

This way, you are up to date instead of looking at a potentially outdated e-mail. I activated e-mail notifications, so you have the best of these two modes.

5.1.10 CAN I SEARCH IN THE EPUB READER?

Yes, consult the Help in your epub readers' documentation.

5.1.11 WHERE DO I FIND THE GITHUB DIRECTORY?

See <https://github.com/cloudmesh-community> and search for your name. It will take some time to set up your repository. A form will be sent to you that asks you to give us your github.com account. Be reminded that we use github.com and not IU GitHub, which has, in the past, been unreliable. Thus, if you do not have an account on github.com, please create one now.

5.1.12 WHERE CAN I FIND THE HID DIRECTORY?

We often refer to the GitHub directory as the hid directory, see *Where to find the GitHub directory?*

5.2 EPUB READERS

This document is distributed in ePub format. Every OS has a suitable ePub reader to view the document. Such readers can, in many cases, also be integrated into a Web browser so that when you click on an ePub, it is automatically opened in your browser. As we use ePubs, the document can be scaled based on the user's preference. If you ever see content that does not fit on a page, we recommend you zoom out to make sure you can see the entire content.

We have made good experiences with the following readers:

- **macOSX:** [Books](#), which is a build in ebook reader
- **Windows 10:** [calibre](#)
- **Linux:** [calibre](#)

If you have an iPad or Tablet with enough memory, you may also be able to use them.

Other browsers are available. Please report if you like one more than the once specified here, so we add your suggestion.

You may want to adjust the zoom of your reader to increase or decrease it. Please adjust your zoom to a level that is comfortable for you. On macOS with a larger monitor, we found that zooming out multiple times results in very good rendering allowing you to see the source code without horizontal scrolling.

6 REFERENCES



- [1] G. von Laszewski, *Cloud computing*, Fall 2019. Bloomington, Indiana: Indiana University, 2019 [Online]. Available: <https://laszewski.github.io/book/cloud/>
- [2] G. von Laszewski, *Python for cloud computing*, Fall 2019. Bloomington, Indiana: Indiana University, 2019 [Online]. Available: <https://laszewski.github.io/book/python/>
- [3] G. von Laszewski, *Linux for cloud computing*, Fall 2019. Bloomington, Indiana: Indiana University, 2019 [Online]. Available: <https://laszewski.github.io/book/linux/>
- [4] G. von Laszewski, *Scientific writing with markdown*, Fall 2019. Bloomington, Indiana: Indiana University, 2019 [Online]. Available: <https://laszewski.github.io/book/writing/>
- [5] G. von Laszewski, *Cloud technologies*, Fall 2019. Bloomington, Indiana: Indiana University, 2019 [Online]. Available: <https://cloudmesh-community.github.io/book/vonLaszewski-cloud-technologies.epub?raw=true>
- [6] G. von Laszewski, “Project format example.” Aug-2019 [Online]. Available: <https://github.com/cloudmesh-community/proceedings-fa18/tree/master/project-report>
- [7] G. von Laszewski, *E516 cloud computing engineering*, Fall 2019. Bloomington, Indiana: Indiana University, 2019 [Online]. Available: <https://laszewski.github.io/book/e516/>