ROI TRAINING
MAXIMIZE YOUR TRAINING INVESTMENT

COURSE NOTES

Course 793B: Programming Microservice and Big Data Applications in the Cloud

ROITRAINING

MAXIMIZE YOUR TRAINING INVESTMENT

Programming Microservice and Big Data Applications in the Cloud

INTRODUCTION

ROI TRAINING

MAXIMIZE YOUR TRAINING INVESTMENT

The following course materials are copyright protected materials. They may not be reproduced or distributed and may only be used by students attending the *Programming Microservice and Big Data Applications in the Cloud* course.

Student Introductions

Please introduce yourself stating:

- Name
- Position or role
- Current project you are working on
- Expectations or a question you'd like answered during this class

Activity: Preparing the Course Activities

Your instructor will divide you into teams

- If your team is in a classroom, you will use flip charts for the activities
- If your team is online, you will use Google Hangouts and Google Slides
  - Every member of the team should exchange their Gmail addresses
  - Using Google Chrome, go to https://hangouts.google.com/
  - One person, click the Video Call icon and invite your team members onto the call
  - Once everyone is on the call, one person should go to https://docs.google.com/presentation/u/0/
  - Click the icon to start a new blank presentation and name it appropriately
  - Lastly, click the Share button on the upper right and invite everyone to edit the presentation
  - Everyone should check their email for the invitation
  - Let your instructor know when everyone has accessed the Slides presentation

About This Course

- This is Part 2 of a three-course series on programming cloud applications
  - 793A: Deploying Applications on Amazon Web Services (AWS) and Google Cloud Platform (GCP)
  - **793B: Programming Microservice and Big Data Applications in the Cloud**
  - 793C: Architecting Applications in the Cloud
- The first course is ***not*** a prerequisite for this course. However, this course assumes some experience using a public cloud like AWS or GCP
- Students can do exercises in this course using AWS or GCP or both

Course Objectives

In this course, we will learn how to:

- Use automation for continuous integration and delivery
- Automate testing
- Design and program microservices using a REST architecture
- Leverage the cloud for Big Data processing
- Enable real-time data analysis
- Add machine learning capabilities to applications

Course Contents

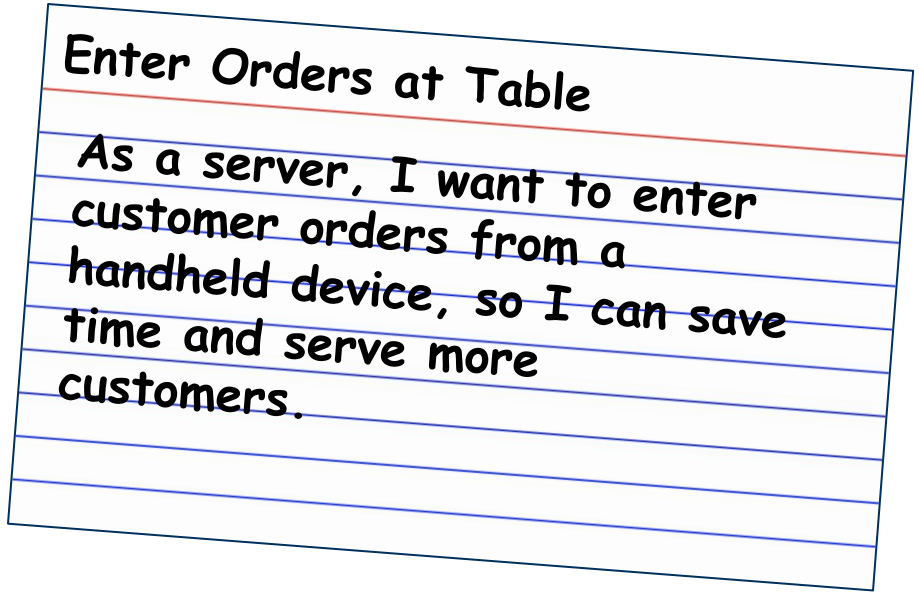
Introducing the Case Study

- Take a few minutes to read the document at the following URL:
  - http://tinyurl.com/793-case-study

Activity: Planning the Case Study

In your teams:

- On index cards, a flip chart, or in Google Slides, write user stories for the MealTime application case study
  - Each story should have a title
  - Use the format: "As a (role), I want to… , so…"
- You should come up with eight or more user stories
- If using Google Slides, write one story per slide



Class Schedule

- Start of class _____
- Morning breaks approximately on the hour
- Lunch _____
- Afternoon breaks approximately on the hour
- Class end _____

ROI's Training Curricula

- Agile Development
- Big Data & Data Analytics
- Business Analysis
- Cloud Computing & Virtualization
- Google Cloud Platform
- ITIL & IT Service Management
- Java 8 / 7
- Leadership & Management Skills
- Microsoft Exchange 2013 / 2010
- .NET & Visual Studio
- Networking & IPv6
- New Hire Programs
- Oracle Database 12*c* / 11*g*
- OpenStack & Docker
- Project Management
- Python & Perl Programming
- Security
- SharePoint
- Software Analysis & Design
- Software Engineering
- SQL Server 2014 / 2012 / 2008 R2
- UNIX and Linux
- Web & Mobile Apps
- Windows 10 / 8.1 / 7
- Windows Server 2012 R2 / 2012 / 2008

*Please visit our website (www.ROItraining.com) for a complete list of offerings.

ROI Training Schedule for Cisco Systems

Course #	Course Title	Course Date
787	DevOps: Digital Transformation for the Cloud & Beyond	Jul 11, 2017
446	TDD and Legacy Code	Aug 1, 2017
787	DevOps: Digital Transformation for the Cloud & Beyond	Sep 6, 2017
103C	Business Analysis	Sep 11, 2017
799	Deploying Infrastructure on Amazon Web Services (AWS)	Sep 11, 2017
756	Practical Security and Cryptography	Sep 11, 2017
798	Developing for Amazon Web Services	Oct 16, 2017
756	Practical Security and Cryptography	Oct 16, 2017
793B	Programming Microservice and Big Data Applications in the Cloud	Oct 17, 2017

ROI Training Schedule for Cisco Systems

Course #	Course Title	Course Date
799	Deploying Infrastructure on Amazon Web Services (AWS)	Oct 17, 2017
793C	Architecting Infrastructure on Amazon Web Services (AWS)	Oct 23, 2017
805C	Foundations of Machine Learning	Oct 23, 2017
793C	Architecting Infrastructure on Amazon Web Services (AWS)	Nov 6, 2017
805C	Foundations of Machine Learning	Nov 6, 2017
793B	Programming Microservice and Big Data Applications in the Cloud	Dec 4, 2017
446	TDD and Legacy Code	Dec 5, 2017

Contact your training coordinator for additional information

ROITRAINING
MAXIMIZE YOUR TRAINING INVESTMENT

Programming Microservice and Big Data Applications in the Cloud

CHAPTER 1: CONTINUOUS INTEGRATION AND DELIVERY

Chapter Objectives

In this chapter, we will:

- Define continuous integration and delivery
- Setup source control and automated builds for continuous integration
- Design a continuous delivery pipeline
- Use tools to automate deployments

Exercise: Setting Up AWS and GCP

- AWS Setup
  https://storage.googleapis.com/roi-code-labs/codelabs/labs/793b-aws-setup/index.html

- GCP Setup
  https://storage.googleapis.com/roi-code-labs/codelabs/labs/793b-gcp-setup/index.html

Chapter Concepts

Source Control

Continuous Integration

Continuous Delivery

Chapter Summary

Continuous Delivery Requirements

- Source Control
- Build Automation
- Test Automation
- Automated Deployment
- Infrastructure
- Visibility and Reporting

Source Control

- Modern source control is a must for CI
- Each developer needs a complete copy of the source
  - Developers should be able to run tests and deploy onto their own machines
- Developers must check into the main source frequently
  - Synchronize the main with their copy prior to checking in
  - Make sure the tests all run and the program builds
- Once checked in, the main source code needs to be built and the tests should be run again

Git

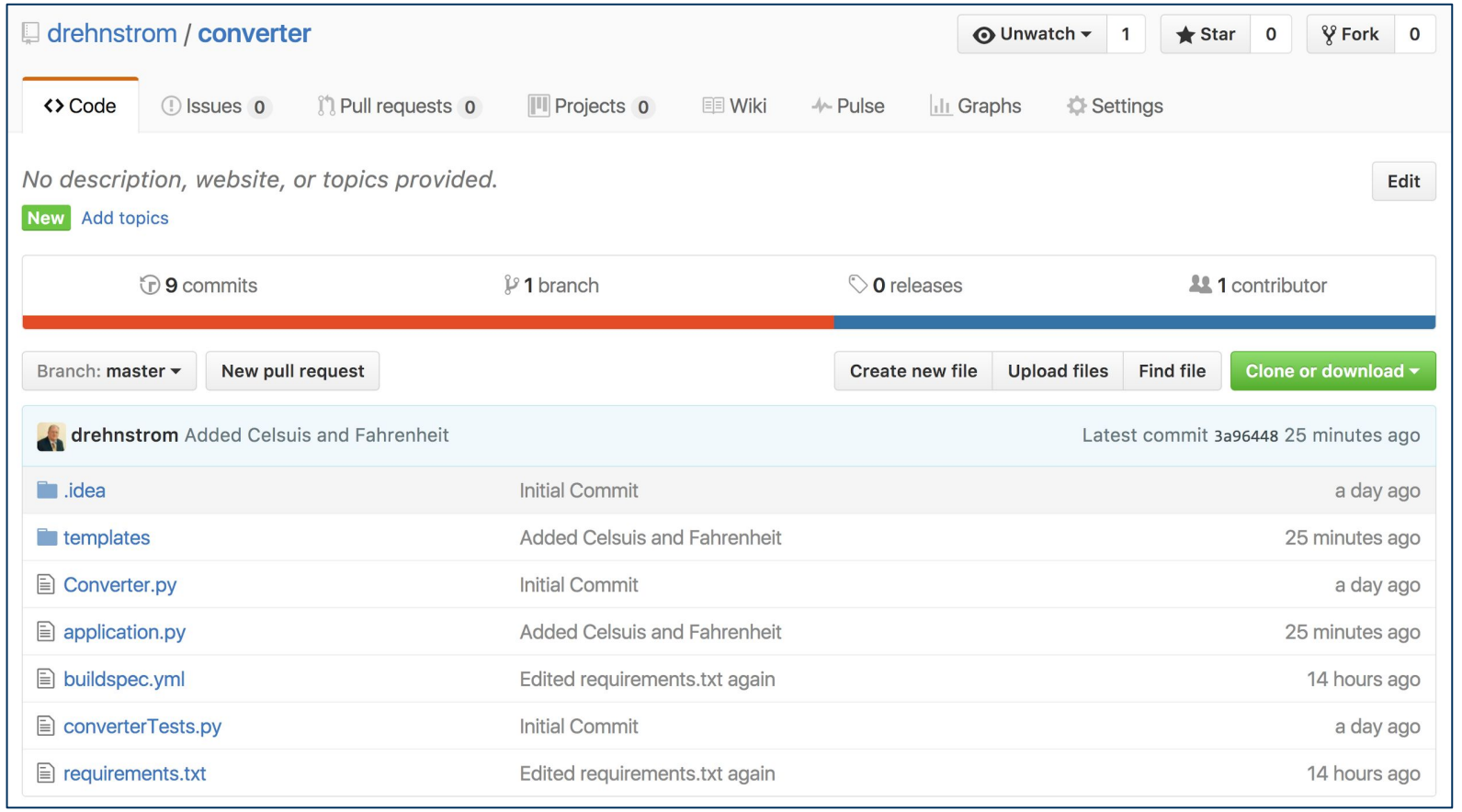
- Free, open-source, distributed source control
  - Create by Linus Torvalds in 2005
  - Used to manage the Linux kernel
  - https://git-scm.com/
- Extremely popular
  - Included on Linux and OSX by default
  - Easy to install on Windows
- Any folder can be a repository
- Many implementations exist
  - https://github.com/
  - Built into cloud systems like AWS and Google Cloud Platform

Some Basic Git Commands

Command	Description
git config --global user.name "Jeff Bezos" git config --global user.email jeff@amazon.com	Configure username and email. Required for commits.
git init	Initialize a local folder as a Git repository. Used for your working folder.
git clone /path/to/repository	Copy a repository.
git add <filename> git add *	Add files to the repository.
git commit -m "Commit message"	Commit changes to the local repository, with a commit message.
git commit -a	Commit and add.
git push origin master	Push local commits to the remote master repository.
git checkout -b <branchname>	Create a new branch and switch to it.
git checkout <branchname>	Switch to a different branch.
git push origin <branchname>	Push a branch to the remote repository.
git pull	Pull changes from the remote repository to your local repository.

GitHub

- GitHub is an online service for sharing Git repositories

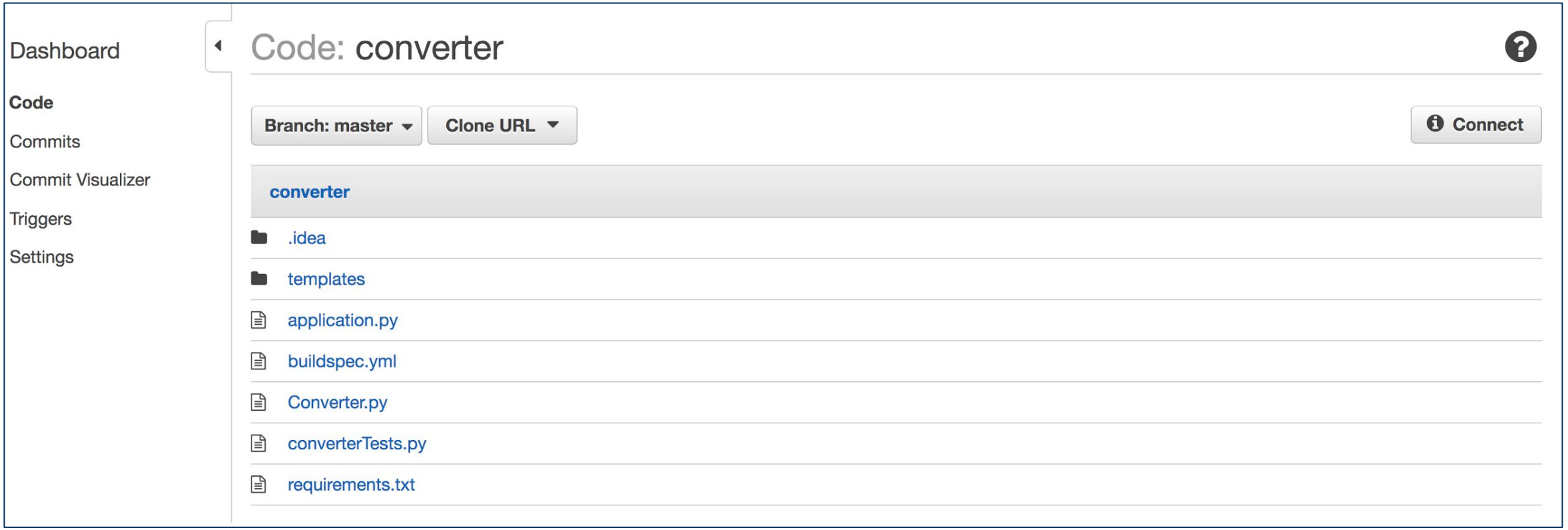


Exercise: Using GitHub

- Run the following tutorial to learn to use GitHub
  - https://guides.github.com/activities/hello-world/
- If it is not already installed, install Git on your local machine
  - Visit https://git-scm.com/
  - Click the Downloads link
  - Follow the instructions for installing Git on your operating system

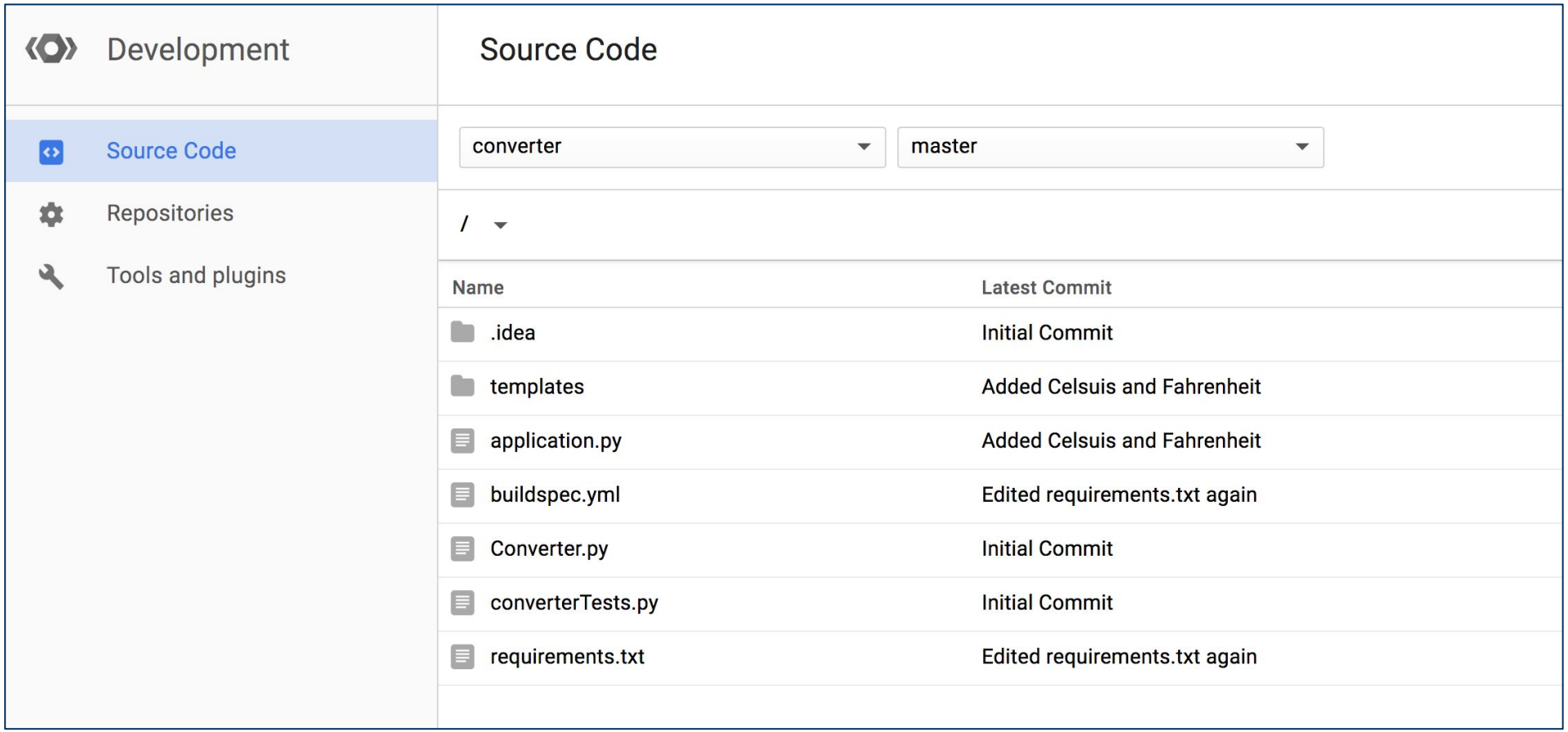
Amazon CodeCommit

- Git service provided by AWS



GCP Cloud Repositories

- Git service provided by Google Cloud Platform



Exercise: Using Git in the Cloud

- AWS CodeCommit
  https://storage.googleapis.com/roi-code-labs/codelabs/labs/793b-aws-codecommit/index.html

- Google Code repositories
  https://storage.googleapis.com/roi-code-labs/codelabs/labs/793b-gcp-repositories/index.html

Chapter Concepts

Source Control

Continuous Integration

Continuous Delivery

Chapter Summary

Continuous Integration (CI) Defined

- Continuous integration is ultimately about automation
  - Tools are used to make testing, quality assurance, and deployment fast and simple
- Deploying new software or new versions should be no big deal
  - Deployments are done frequently to test environments
  - Deployment to production should be as simple as a configuration change
- CI is a long-term goal and takes time and effort to achieve

> *"Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily—leading to multiple integrations per day."*
>
> —Martin Fowler
>
> http://www.martinfowler.com/articles/continuousIntegration.html

Build Automation

- Ideally, builds should be automatic and run as soon as code is checked in
  - Little or no human intervention should be required to run a build
- Builds should include:
  - Compilation
  - Running of automated tests
  - Running of code quality tools
  - Reporting of build quality
  - Deployment of final product to a test environment

Build Automation Tools

- Build automation tools are used to:
  - Compile source code
  - Download and update dependencies
  - Execute automated tests
  - Package build output
  - Deploy
  - Etc.
- In a DevOps environment, manual compilation, testing, and deployment is not efficient enough

Apache Maven

- Build automation tool for Java with plugins for other languages as well
  - Describes how the software is built
  - Manages project dependencies and makes sure they are up-to-date
  - https://maven.apache.org/
- Uses XML in `pom.xml` file to describe how the project is built
  - Example `pom.xml` file:

```
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0</version>
  <dependencies>
    <dependency>
          <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Gradle

- Open source build automation tool
  - Built in Java but works with any language and integrates with many tools
  - Builds on Ant and Maven
  - https://gradle.org/
- Uses a domain-specific language based on Groovy for build scripts
  - Groovy is a similar to Python and Ruby, but compiles to JVM bytecode
  - Example build script:

```
group 'com.me.app'
version '1.0'
apply plugin: 'java'

sourceCompatibility = 1.5

repositories {
    mavenCentral()
}
dependencies {
    testCompile group: 'junit', name: 'junit', version: '4.11'
}
```

Test Automation

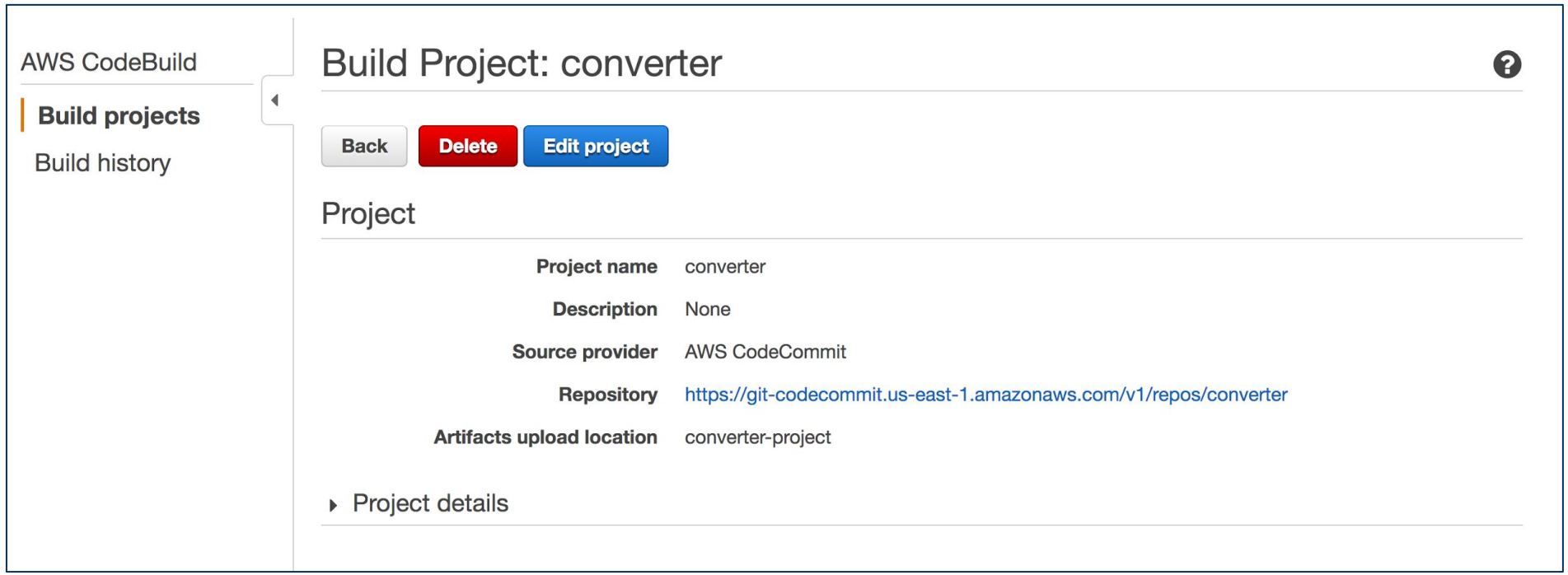
- Automated tests should be run every time a build happens
  - Whether on the server or the developers' machines
- Automated tests should include:
  - Unit tests
  - Integration tests
  - System tests
  - End-to-end tests

Self-Study: What Is Continuous Integration?

- Take a few minutes and read the article at the following URL:
  - http://www.martinfowler.com/articles/continuousIntegration.html

AWS CodeBuild

- Code Build pulls source code from a repository, executes a set of build commands, and tests and writes results to some output location



Defining AWS CodeBuild Builds

- Builds are defined in the `buildspec.yml` file
  - Essentially a list of commands and a list of files to copy

buildspec.yml

```
version: 0.1

phases:
  build:
    commands:
      - echo Converter build started on `date`
      - echo Compiling the code...
      - echo Running Tests...
      - python converterTests.py
  post_build:
    commands:
      - echo Hey, Build completed on `date` woo hoo!
artifacts:
  files:
    - application.py
    - Converter.py
    - converterTests.py
    - requirements.txt
    - templates/index.html
    - templates/layout.html
    - templates/temp_converter.html
```

Exercise: Continuous Integration

- AWS CodeBuild
  https://storage.googleapis.com/roi-code-labs/codelabs/labs/793b-aws-codebuild/index.html

Chapter Concepts

Source Control

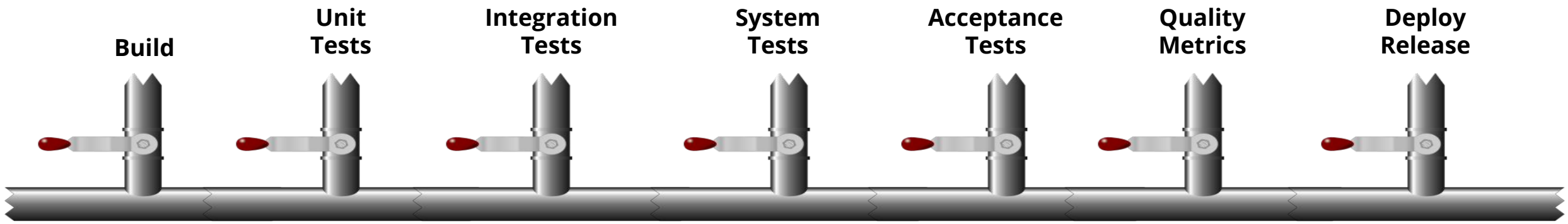
Continuous Integration

Continuous Delivery

Chapter Summary

Continuous Delivery (CD) Defined

- Releases into production are automated based on quality metrics
  - Software is released whenever it meets the quality standards
- Small releases happen very frequently
  - Less risky than big-bang releases for both customers and developers
- Continuous delivery pipeline is used to manage the deployment process
  - Steps are defined within the pipeline
  - Some quality metric is used to trigger the next step
  - The final step is deploying a new release into production



Continuous Delivery and Continuous Integration

- Continuous delivery is a next step after continuous integration
- It is an evolutionary process
  - Need automated tests, builds, and source control to do CI
  - Need repeatable, dependable CI environment achieve CD

Automated Deployment

- Whatever the final output of the software is, that should be created with every build
  - If the final output is an installation program, then create that program
  - If the final output is a website or service, deploy it to a test server
- Should be ready to go live at any time if required
- The day the software goes live should not be stressful

Infrastructure as Code

- Infrastructure as code uses scripts to automate server configuration
  - New servers just rerun the script and configure themselves
- Easier to create for new machines
- Don’t apply patches, instead create a new server with the latest software and rerun the script
- Perfect for virtualized environment
  - VMs are just standard machines which use the script for configuration

Chef

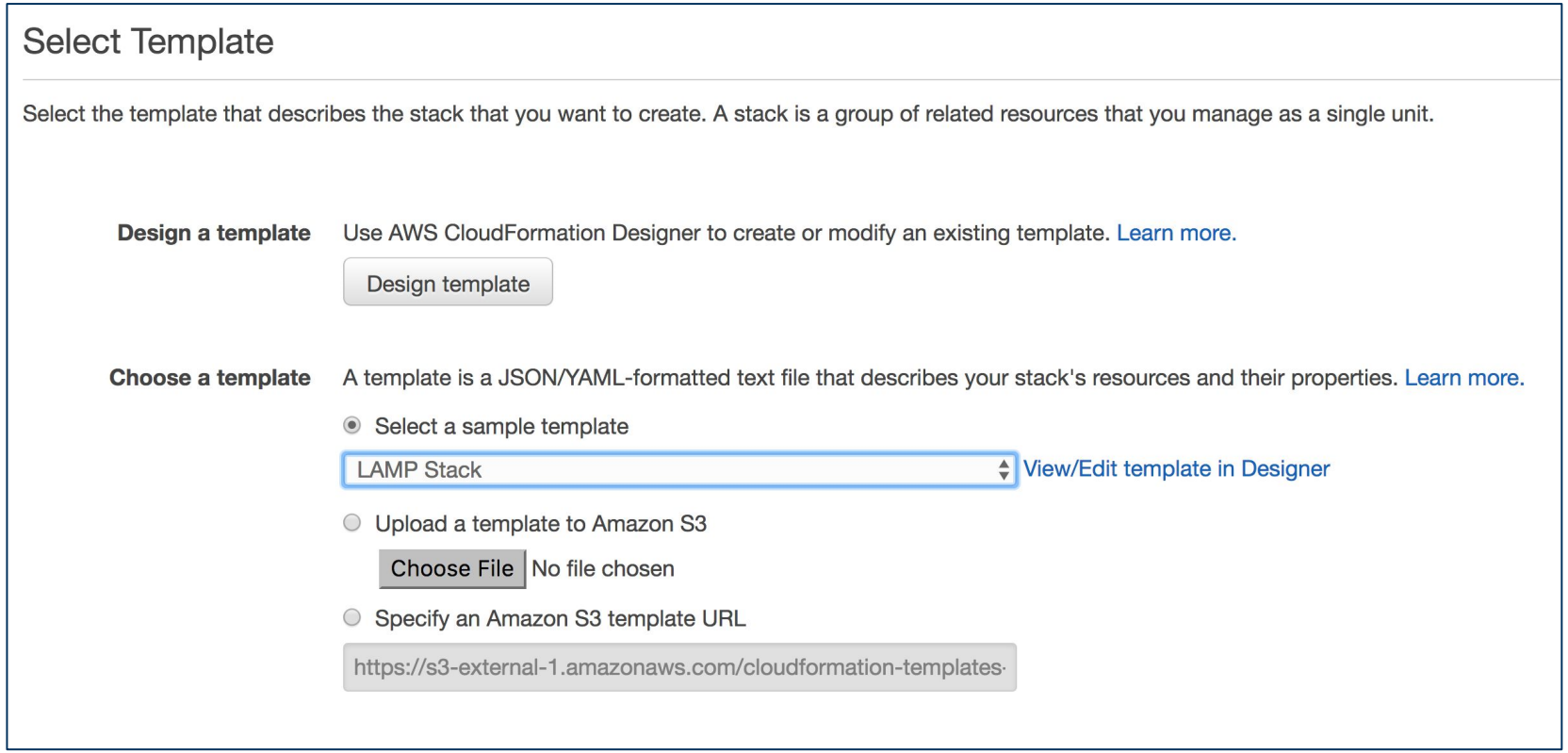
- Used to programmatically configure servers using scripts
  - Scripts are written in a domain-specific language similar to Ruby
  - https://www.chef.io/
- Can be used with any kind of server, Linux, or Window
- Scripts use a recipe metaphor and are used to:
  - Install software (web servers, database servers, etc.)
  - Run updates and install patches
  - Deploy source code
- Multiple recipes are managed in a cookbook

Puppet

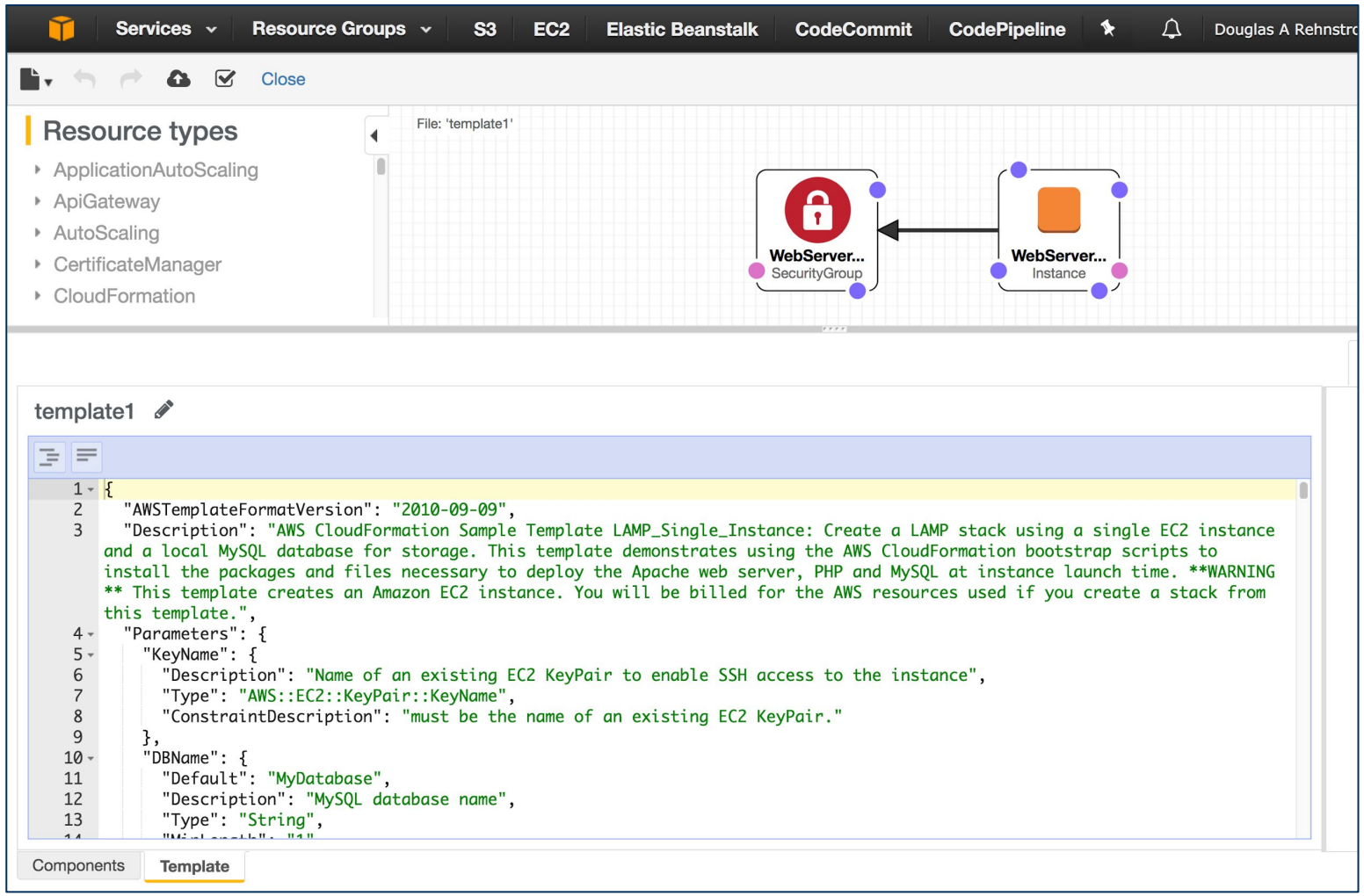
- Uses a model approach to defining infrastructure
  - JSON-like syntax is used to define infrastructure
    - Chef uses a scripting approach
  - Puppet knows how to build the infrastructure based on the model
  - https://puppet.com/

AWS CloudFormation

- Uses JSON templates to define environments
  - Environments consist of one or machines, networks, databases, etc.

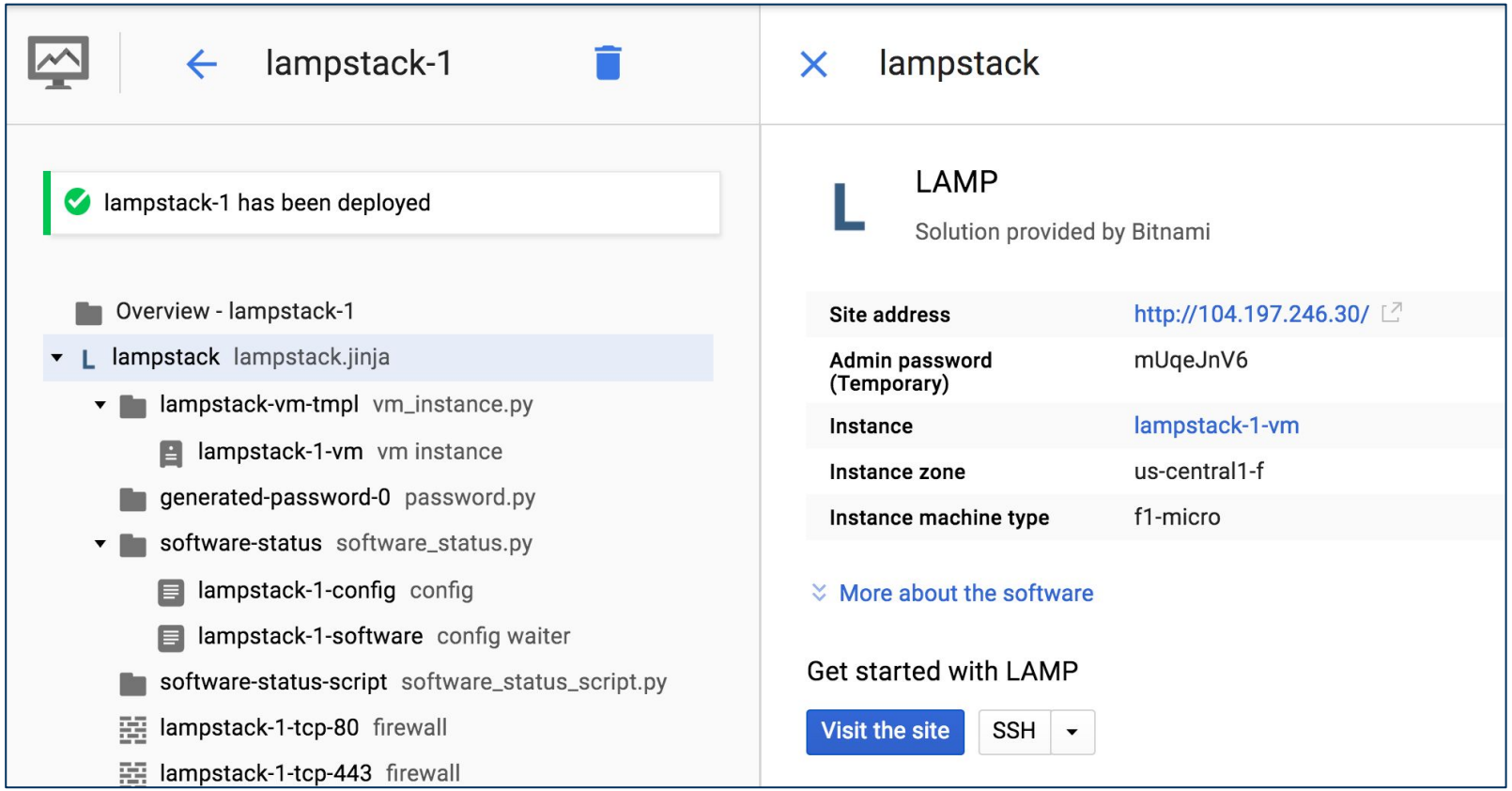


Example CloudFormation Template



GCP Deployment Manager

- Uses YAML or JINJA templates to define environments



Example Deployment Manager Template

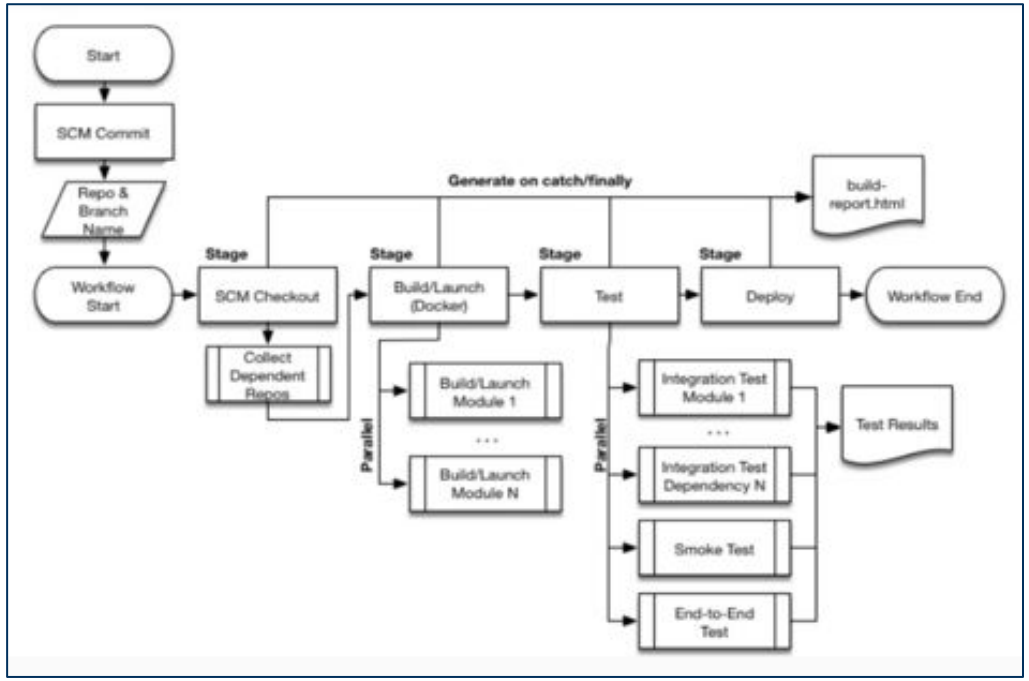
```
{% import "path_utils.jinja" as path_utils with context %}

{% set project = env["project"] %}
{% set deployment = env["deployment"] %}
{% set name = "%s-vm-tmpl" % env["name"] %}
{% set instanceName = "%s-vm" % deployment %}
{% set zone = properties["zone"] %}
{% set machineType = properties["machineType"] %}
...

resources:
  - name: {{ name }}
    type: vm_instance.py
    properties:
      instanceName: {{ instanceName }}
      sourceImage:
https://www.googleapis.com/compute/v1/projects/bitnami-launchpad/global/images/bitnami-lampstack-5-6-30-1-linux-debian-8-x86-64
      zone: {{ zone }}
      machineType: {{ machineType }}
      network: {{ network }}
      {% if subnetwork %}
      subnetwork: {{ subnetwork }}
      {% endif %}
...
```

Continuous Integration Servers

- Continuous Integration servers are used to:
  - Monitor a version control repository
  - Start an automated build when source code is checked in
  - Report on build history and build quality
  - Manage other tasks as required
- CI servers are used to manage a continuous delivery pipeline

Jenkins

- Open-source automation server for managing build and deployment tasks
  - https://jenkins.io/
  - Is a Java Web application
- Create a pipeline which defines the workflow of a build
  - Monitor source control
  - Check out latest source
  - Compile
  - Run tests
  - Deploy
  - Report on results

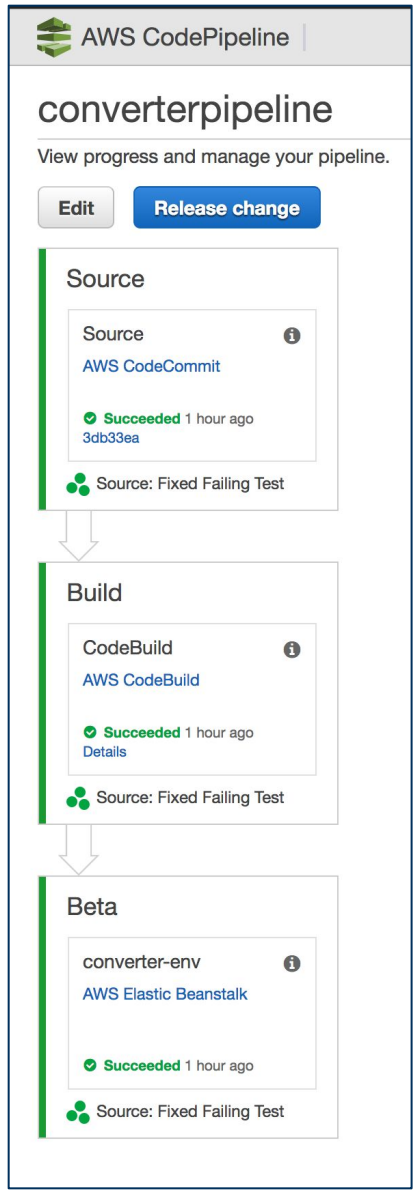


Team Foundation Server

- TFS can be used as a continuous integration server
  - Supports TFVC and Git source control repositories
  - CI Builds can be defined that execute when source code is checked in
  - Tests can be integrated into the builds
  - Can define automated deployment tasks
- Ideal for Windows environments and Visual Studio development teams
  - Especially when TFS is also being used to manage work

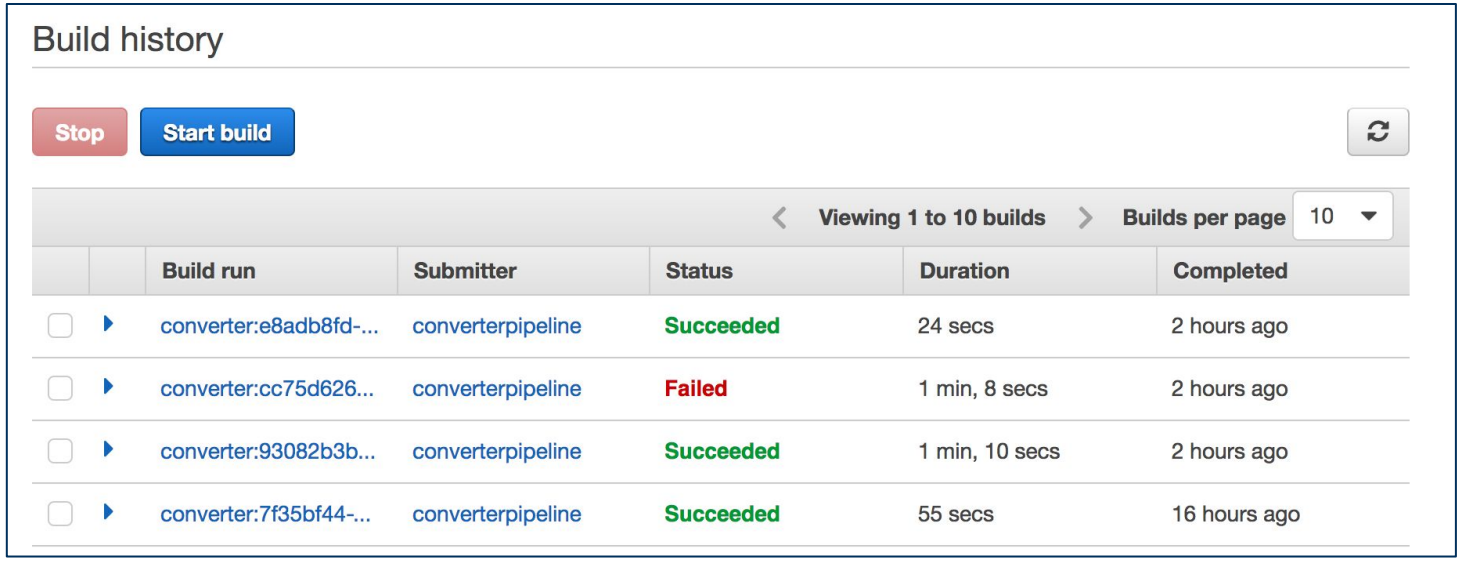
AWS CodePipeline

- Defines a deployment as a series of steps
  - Watches a repository waiting for a change
  - When the repository changes, perform a build
  - If the build succeeds, do a deployment
- Supports Amazon CodeCommit or GitHub repositories
- Can deploy to various deployment platforms including Elastic Beanstalk, Cloud Formation, and others



Visibility and Reporting

- All stakeholders and team members should see CI progress
- Make available via some website:
  - Results of builds (success or failure)
  - Test results
  - Code quality metrics
  - History of builds
- Can be used to determine when to go to production



Exercise: CodePipeline

- AWS CodePipeline
  https://storage.googleapis.com/roi-code-labs/codelabs/labs/793b-aws-code-pipeline/index.html

Chapter Concepts

Continuous Integration

Continuous Delivery

Infrastructure

Chapter Summary

Chapter Summary

- Automated builds should run when changes are pushed to the code repository
- Automated tests should be a part of a build
- A continuous delivery pipeline can deploy applications if tests all pass and the build succeeds
- Tools should be used to automate testing, deployment, configuration, prerequisites, etc.

ROITRAINING
MAXIMIZE YOUR TRAINING INVESTMENT

Programming Microservice and Big Data Applications in the Cloud

CHAPTER 2: TEST AUTOMATION

Chapter Objectives

- Define the types of tests that must be done to complete software
- Write Unit tests
- Explore how to write end-to-end automated tests

Chapter Concepts

Test-Driven Development

Unit Testing

Integration and End-to-End Testing

Chapter Summary

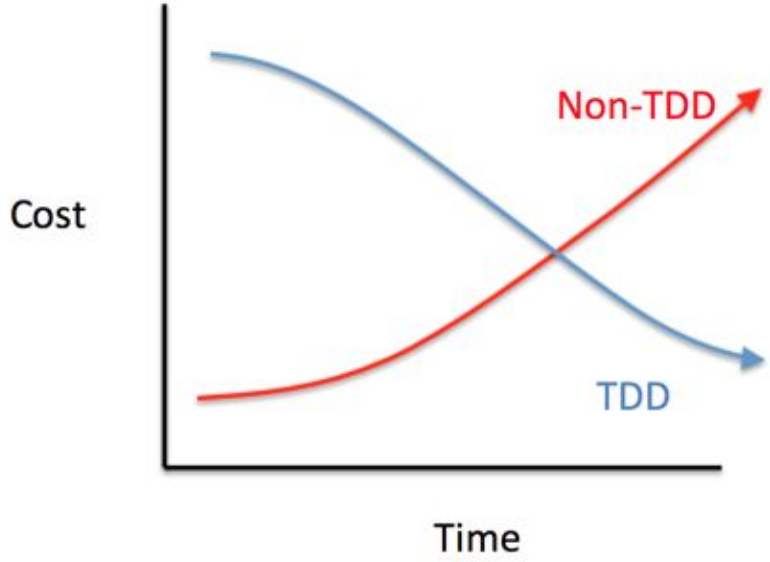
Test-Driven Development

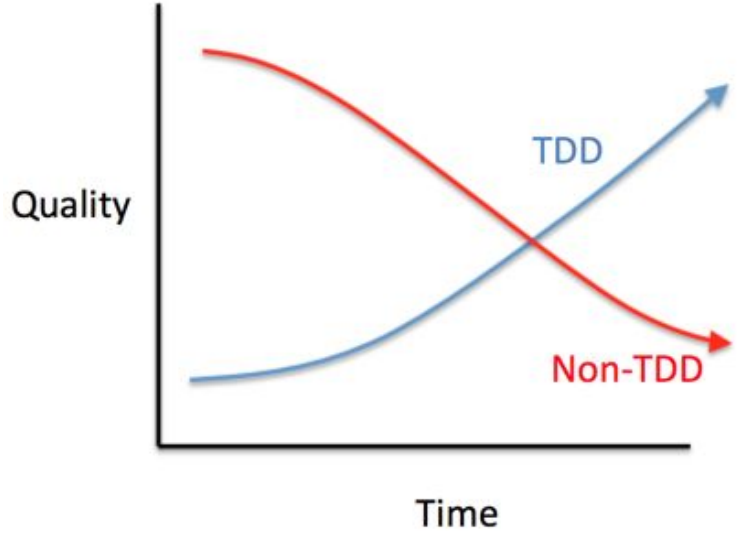
- Traditional developments put testing toward the end
  - Creates a mismatch between requirements and testing
  - Removes responsibility of testing from the programmers
- Test-driven developments move testing to the beginning
  - Testing is done continuously
  - The entire team becomes responsible for testing
- Constant testing helps with everything the team does
  - Tests help verify the requirements
  - Tests simplify design and architecture decisions
  - Tests help programmers know their code is bug free
  - Tests build confidence of customers and users

Benefits of TDD

- Over time, the cost of maintaining software is lower when there are thorough suites of automated tests
- With big upfront design processes, design quality starts high, but tends to degrade over time
- With TDD, testing is continuous and quality increases over time
  - "Constant attention to technical excellence increases agility"

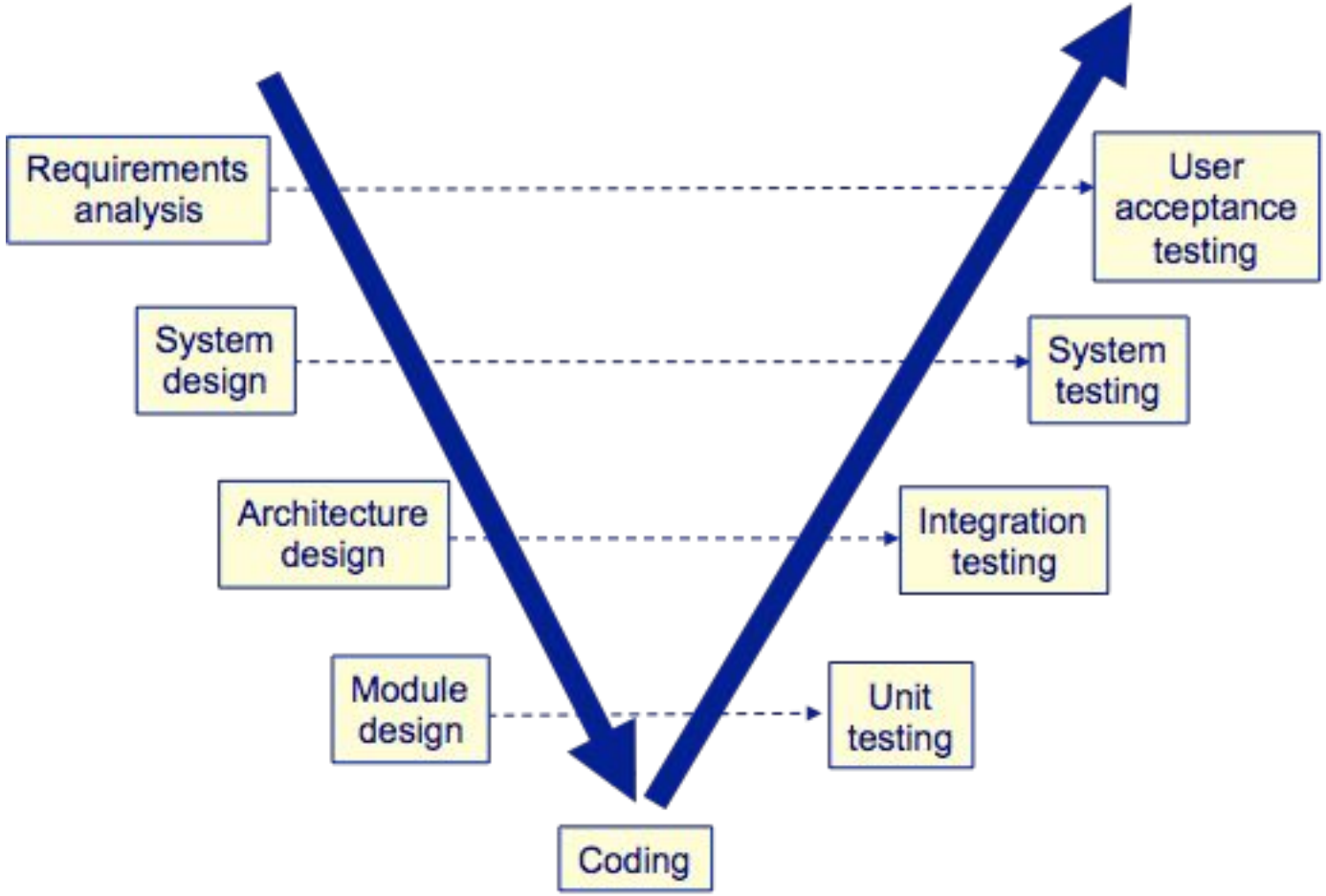
—Agile Manifesto





The V-Model of Testing

- There are different types of tests for the things development teams work on



Test Types Defined

- *Unit tests* are low-level tests that prove that all classes work as expected
  - Each class has a corresponding test class, called a test fixture
  - Every function provided by the class has one or more test functions
- *Integration tests* prove that classes work together
  - When classes use other classes, does everything work?
  - Do classes work with other aspects of the system (files, databases, etc.)?
- *System tests* prove that an application works when deployed onto a system
  - System tests include load testing, security testing, and black box testing
- *User acceptance tests* prove that a program does what users expect it to do
  - Test can be automated or scripted based on use cases
  - Test can be exploratory

Chapter Concepts

Test-Driven Development

Unit Testing

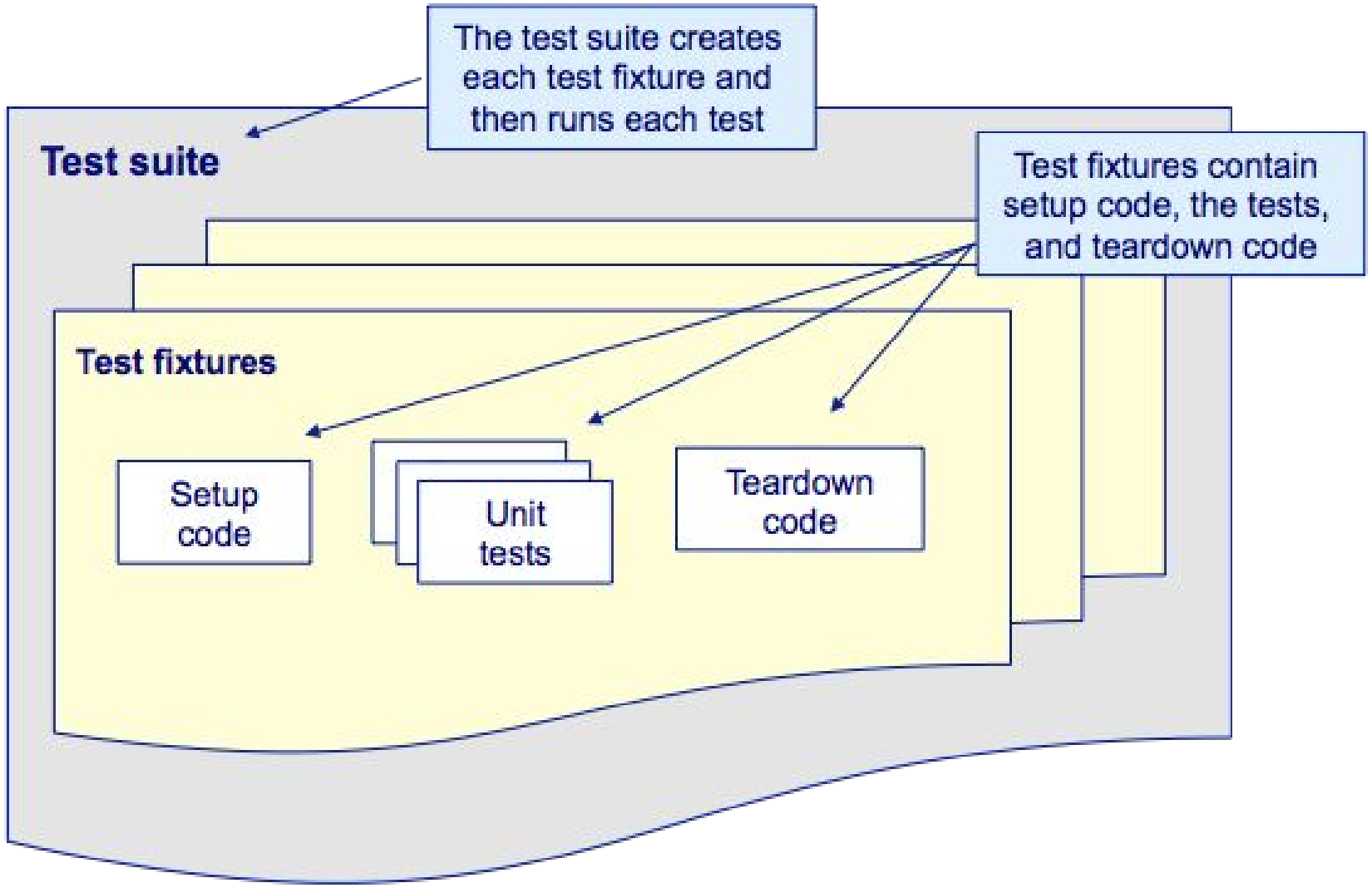
Integration and End-to-End Testing

Chapter Summary

Unit Tests

- Are automated tests that prove a single “unit” of code works as expected
  - Each unit has a corresponding test fixture
  - Each behavior within a unit has one or more test methods
  - All external behaviors need tests (constructors, methods, functions, properties, exceptions, etc.)
- What a unit is, depends on the language
  - In object-oriented languages (e.g., Java or Python), a unit is a class
  - In procedural languages (e.g., C), a unit might be a module
  - In functional languages (e.g., JavaScript), a unit might be a function
  - A microservice might be treated as a unit in a service-oriented program

Unit Testing Components



Programming Test Fixtures

- Each test fixture creates instances of the class being tested
  - Class must be instantiated before each test
  - Ensures one test doesn't affect another
- Setup code creates and initializes the class being tested
  - Ran before each individual test
  - May set preconditions of the class being tested, if required
- Test methods test each property and method of the class being tested
  - Each test should end with an assertion
  - If the assertion is true, the test passes, otherwise it fails
- Teardown code cleans up after the tests run
  - Useful when database records or files are being created in tests and should be deleted when the tests are complete

Test Fixture Example Pseudocode

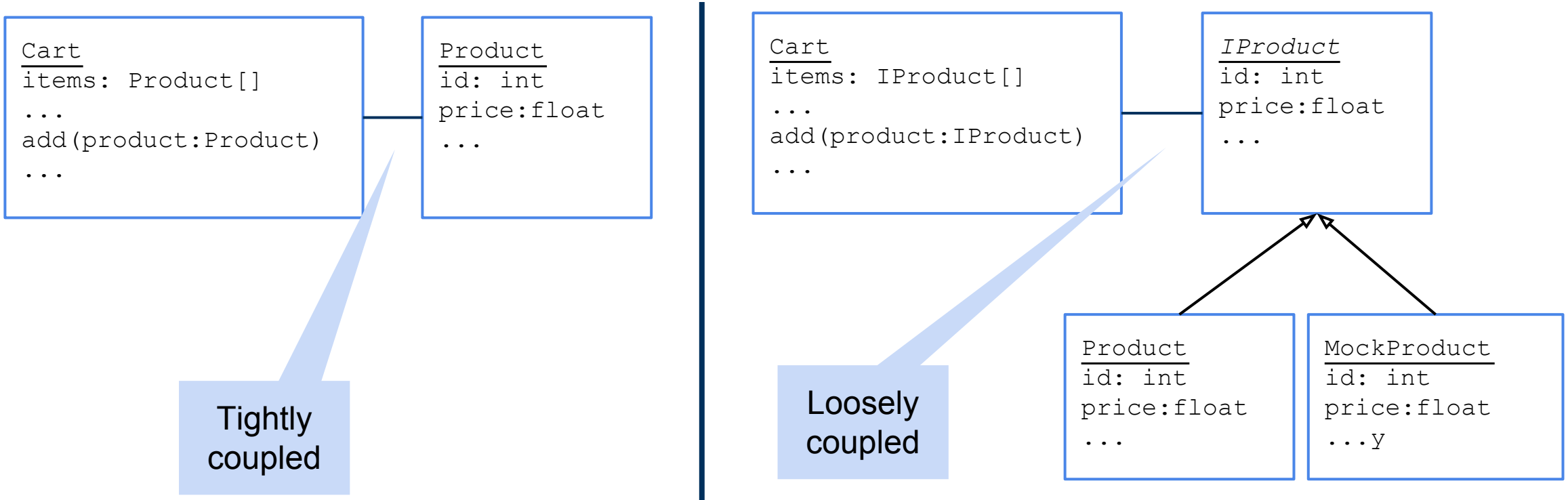
```
class CartTests

    testObj as Cart

    setup()
        testObj = new Cart()

    constructorTest()
        assert(testObj != null)

     countTest()
        assert(testObj.count == 0)

    addTest()
        testObj.add(new MockProduct())
        assert(testObj.count == 1)

    ...
```

Isolating Classes for Testing

- Classes must be tested in isolation
  - Dependencies must be mocked
  - Must prevent a bug in a dependency from causing a unit test to fail
- Mock objects look like the real thing but fake real behaviors
  - Always return predictable, consistent results
  - Classes, databases, and services are all things that need to be mocked
- Isolating classes forces loosely-coupled architectures
  - Proper testing and TDD lead toward good designs
  - Software that is easy to test, is also easy to use and easy to maintain

Designing for Mock Objects

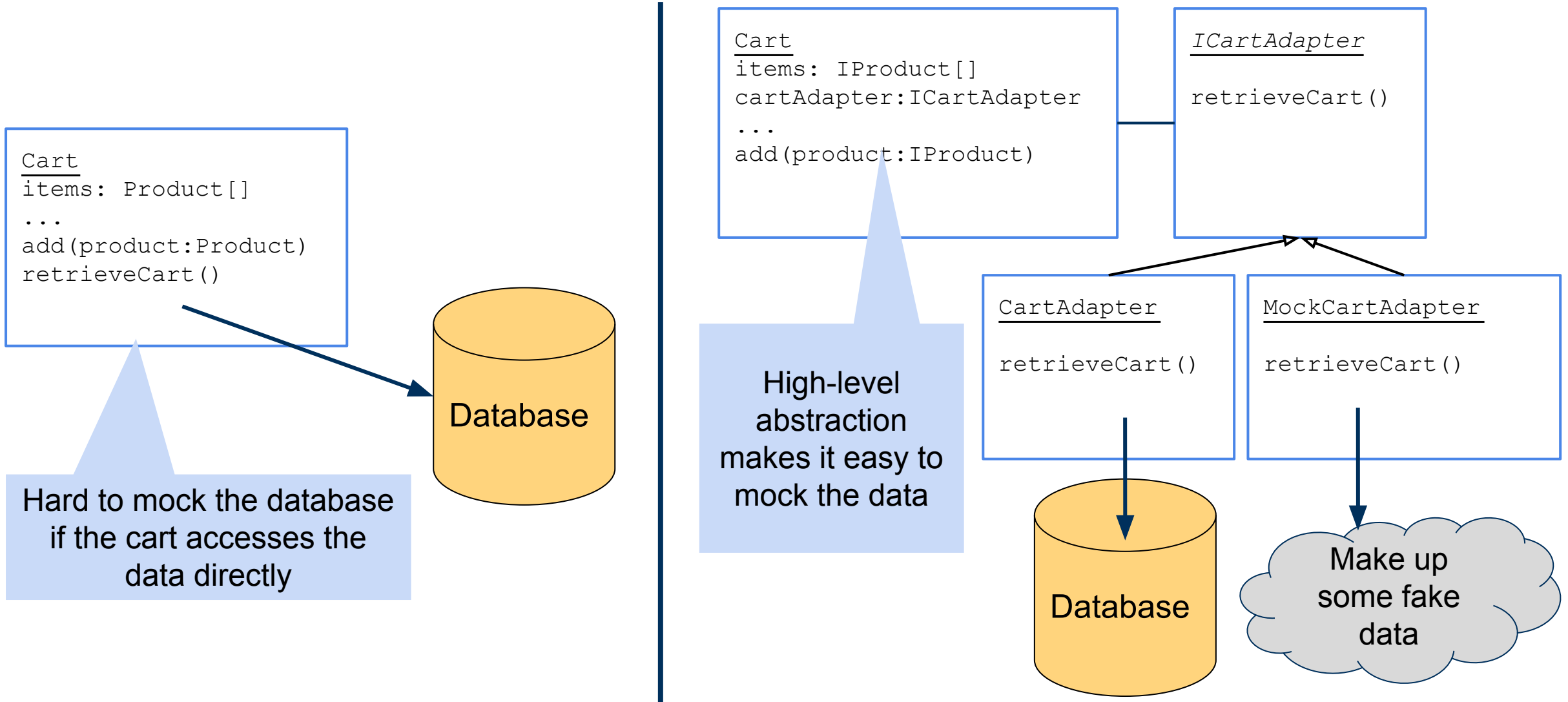
- Objects should hold references to abstract not concrete types
  - Allows mock objects to be created easily
  - Software is more flexible and easier to maintain



Dependency Inversion Principle

- A high-level class should not depend on a low-level class
  - The high-level class should reference some abstraction
  - The abstraction should be defined according to the needs of the high-level class, not in terms of the behaviors of the low-level class
- Inverting the dependencies makes it easier to mock external systems like databases and services
- Leads to a more flexible architecture
  - Easier to test
  - Easier to maintain
  - More accommodating to change

Dependency Inversion Illustrated



Dependency Injection

- Objects should not be responsible for creating their own dependencies
  - Objects should ask some other entity to give them dependencies
- Project being tested should not be aware of mock objects
  - Test projects should determine which mock objects should be used
  - Dependency injection allows the test project to insert mocks as needed
- Requires a factory method to create the objects and configuration code
  - Can use reflection or a dependency injection framework

Dependency Injection Pseudocode

```
class CartAdapterFactory

   getCartAdapter()
      cartAdapterType = system.configuration.settings["cartAdapter"]

      Class theClass = Class.forName(cartAdapterType);
         return  (CartAdapter)theClass.newInstance();
```

Mocking and Dependency Injection Frameworks

- Teams can use various frameworks that automate the creation of mock objects and dependency injection
  - Some might argue that well-designed systems don't need these frameworks
- Dependency injection frameworks include:
  - Spring
  - Google Guice
- Mocking frameworks include:
  - Mockito
  - jMock
  - EasyMock

Unit Testing Tips

- Three laws of unit testing:
  - Don’t write production code until you have a failing test
  - Don’t write more of a unit test than is sufficient to fail
  - Only write enough code to get the tests to pass
- Keep test code clean and refactored
- Avoid multiple asserts per test
- Tests should only test one thing
- Tests should be fast
- Each test should be independent of any others (*order doesn’t matter*)
- Test all public behaviors

Activity: Unit Testing

In your teams:

- On a flip chart or in Google Slides, choose one of your user stories
- In pseudocode, write a unit test fixture as shown in the course slides
  - Try to come up with a number of tests
  - You can write them on paper, a flip chart, or electronically, whichever you prefer
- BONUS: Go to your MealTime project and start writing some real tests
  - Make sure the Code Pipeline works
  - Check in some failing tests to see the Code Pipeline fail

Chapter Concepts

Test-Driven Development

Unit Testing

Integration and End-to-End Testing

Chapter Summary

Integration Tests

- *Unit tests* ensure individual classes work in isolation of one another
- *Integration tests* prove classes and components still work when put together
  - Classes within projects need to work together
  - Websites need to work with the components that they use
  - Components need to be able to successfully read and write from storage
- *End-to-end tests* are automated tests that control the entire system
  - From UI through to storage
- Automated tools exist for creating integration and end-to-end tests
  - Can sometimes simply use the unit testing framework
    - Limited ability to control user interfaces
  - Other testing tools exist to script interaction at the UI

Selenium

- Is a tool for automating browsers and their interaction with websites
  - Works in all major browsers
  - Includes an IDE that allows for creating scripts
- Selenium Web Driver is an API that programmatically controls browsers
  - Can be used with many languages (Java, Python, JavaScript, etc.)
  - Integrates with common unit testing frameworks (JUnit, PyUnit, etc.)
- Allows automated tests to be created to control the browsers
  - Request pages
  - Fill in forms
  - Submit forms
  - Search page for expected results

Steps for Using Selenium

1. Specify a browser
2. Request a page
3. Fill out a form
4. Submit a request
5. Pause (*give the system time to get the response*)
6. Assert something is true about the resulting page

Using Selenium Pseudocode

```
class SearchPageTests

    constructor()
          SeleniumWebDriver.Bootstrap(
               SeleniumWebDriver.Browser.Chrome)

     openSearchPageTest()
          I.Open("http://tests.mysite.com/search");
            I.Assert.Exists("#searchTextBox");

     searchForProductTest()
          I.Open("http://tests.mysite.com/search");
            I.Enter("Computers").In("#searchTextBox");
            I.Click("#submit");
            I.Wait(1);
           I.Assert.Text("Displaying results for Computers").In("#resultsMessage");
```

Chapter Concepts

Test-Driven Development

Unit Testing

Integration and End-to-End Testing

Chapter Summary

Chapter Summary

- There are different types of tests for all levels of software development
- Unit tests are low-level tests that verify code works as expected
- End-to-end tests are used to verify components work when they are integrated together

ROITRAINING

MAXIMIZE YOUR TRAINING INVESTMENT

Programming Microservice and Big Data Applications in the Cloud

CHAPTER 3: BUILDING MICROSERVICES

Chapter Objectives

- Define microservice architectures
- Design an application that uses microservices
- Program RESTful services
- Leverage Docker for microservice deployment

Chapter Concepts

Microservice Architecture

Programming Microservices

Deploying Microservices with Docker

Chapter Summary

Software Architecture

- A program's architecture consists of its collection of components, modules, classes, and functions
- The goals of architecting a system include:
  - Simplicity
  - Ease of maintenance
  - Reusability
  - Efficiency
  - And others
- Architectural choices involve trade-offs
  - Dividing large functions into smaller ones decreases code complexity, but increases architectural complexity

Service-Oriented Architecture (SOA)

- Divides a program into multiple, composable services
  - Services can communicate with one another over a network
- Services are language agnostic
- Services can be composed in different ways
  - Promotes reuse
  - Allows programmers to create new uses for existing services
- Early web-based services used an RPC protocol called SOAP
  - Messages promoted tight-coupling to the messages
  - Message format in XML
  - Unnecessarily complex for many web-based applications

Microservice Architecture

- Similar to service-oriented architecture
  - Divides large programs into small, loosely-coupled components
  - Components communicate via HTTP(s)
- Microservices should be small enough for a single team to own completely
  - Services should perform a single job
- Microservices use an architectural style called REST
  - Commonly uses simple HTTP get, post, put, and delete requests
  - Data is passed between services using a textual encoding such as JSON
  - Very lightweight communication protocol
- Promotes loose coupling and composition

Advantages of Microservices

- Services can be programmed independently of one another
  - Each team is responsible for one or more services
  - A team is responsible for all aspects of a service they own
- Services can be deployed and run independently of one another
  - No need for large, complex deployments
  - Reduces the risk of introducing new features
- Services can scale independently of one another
  - Busy services get more resources
- Different services can be programmed in different languages or frameworks
  - Programmers choose the tools that best achieve the service goals
- Microservices allow for easier innovation
  - Large, monolithic programs are difficult to change and risky to deploy

Microservices

From Wikipedia, the free encyclopedia

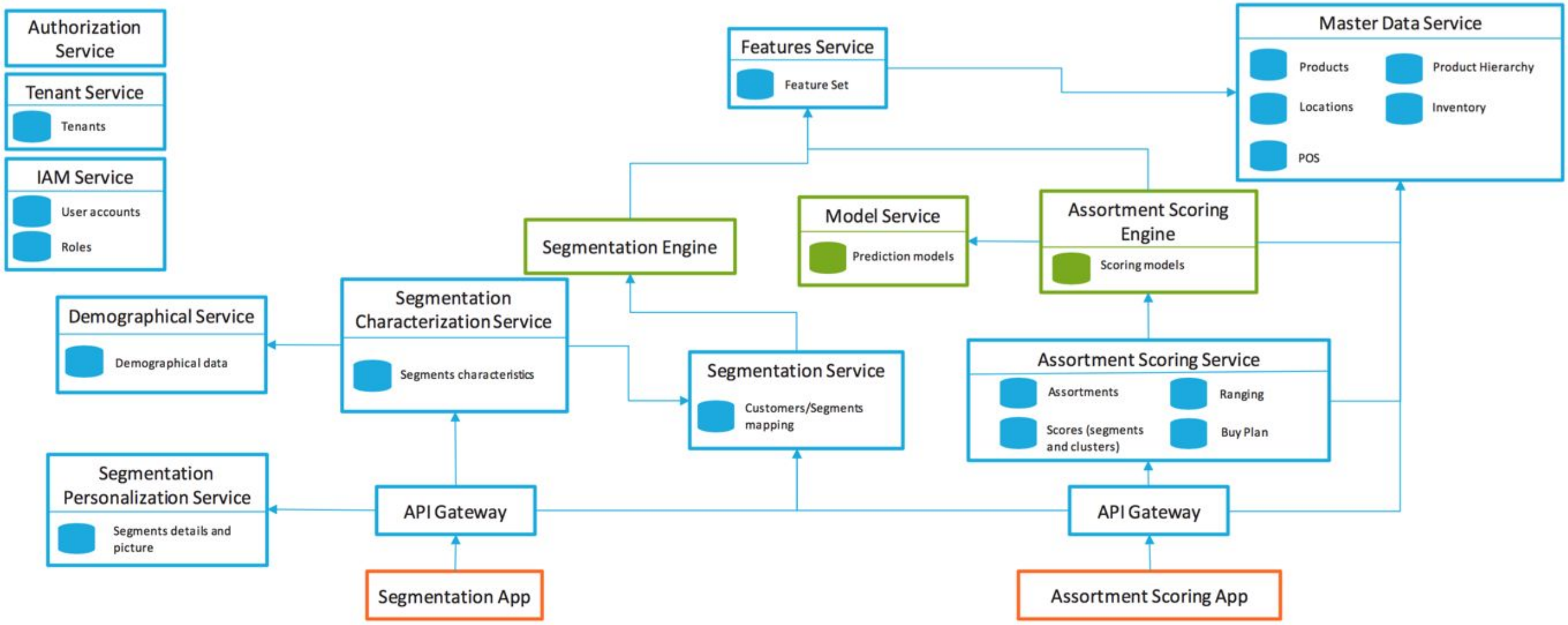
Microservices are a more concrete and modern interpretation of service-oriented architectures (SOA) used to build distributed software systems. Like in SOA, services in a microservice architecture are processes that communicate with each other over the network in order to fulfill a goal. Also, like in SOA, these services use technology agnostic protocols.[1] Microservices architectural style is a first realisation of SOA that has happened after the introduction of DevOps and this is becoming the standard for building continuously deployed systems.[2]

In contrast to SOA, microservices gives an answer to the question of how big a service should be and how they should communicate with each other. In a microservices architecture, services should be small and the protocols should be lightweight. The benefit of distributing different responsibilities of the system into different smaller services is that it enhances the cohesion and decreases the coupling. This makes it much easier to change and add functions and qualities to the system anytime.[3] It also allows the architecture of an individual service to emerge through continuous refactoring,[4] hence reduces the need for a big up front design and allows for releasing the software early and continuously.

Designing Microservices

- Single Responsibility Principle
  - Microservices should perform one job completely
  - The functions of a microservice should all seek to fulfill the service goal
    - Services with many unrelated functions lack cohesion
- Microservices should be testable
  - If a service is hard to test, it is likely hard to use and hard to maintain
- Microservices should be loosely coupled
  - If two services need to know too much about each other, maybe they should be combined
- Microservices should not share state
  - There should not be a huge, all-encompassing master database

Microservice Architecture Example



Activity: Diagramming Microservices

- Our company has decided to build an online sales platform
  - Includes a retail storefront, administration, catalog management, warehouse integration, payment processing, etc.
- Fifty developers will work on this project, but we are going Agile and have divided them into seven different teams
  - Each team needs to be responsible for their own services
  - The goal is for teams to be as independent as possible
- In your groups, do a high-level initial diagram of the services required by the application
  - Draw the diagram on flip charts
  - Use the prior slide as a guide

Chapter Concepts

Microservice Architecture

Programming Microservices

Deploying Microservices with Docker

Chapter Summary

REST

- Representational State Transfer
  - An architectural style
  - Scalable up to WWW and down to micro-services
- Protocol independent
  - HTTP is most common
  - SMTP could be used
  - Others possible
- Service endpoints supporting REST are called RESTful
- Client and Server communicate with Request – Response processing

Features of a RESTful Service

- URI identifies a Resource (an Endpoint)
- Responses return an Immutable Representation of the Resource information
- Uniform interface
- Stateless
- Representation can have Links to additional Resources
- Caching of Immutable Representations

Resources and Representations

- Resource is an abstract notion of information
- Representation is a copy of the Resource Information
- Representation can be a single item of a set (or bag)

Design Steps

- Identify the resources
- Design identifiers
- Select set of Representational forms
  - JSON
  - XML
  - HTML
  - …

```
<Person>

<ID>1</ID>

<Name>M Vaqqas</Name>

<Email>m.vaqqas@gmail.com</Email>

<Country>India</Country>
</Person>
```

```
{
    "ID": "1",
    "Name": "M Vaqqas",
    "Email": "m.vaqqas@gmail.com",
    "Country": "India"
}
```

HTTP Request

- <VERB>: method: GET, PUT, POST, DELETE, OPTIONS
- <URI>: Uniform Resource Identifier (name)
- <Request Header>: metadata about the message
  - Preferred Representation formats (e.g., JSON, XML, etc.)
- <Request Body>: (Optional) Request state
  - Representation (JSON, XML) of resource

HTTP Request Examples

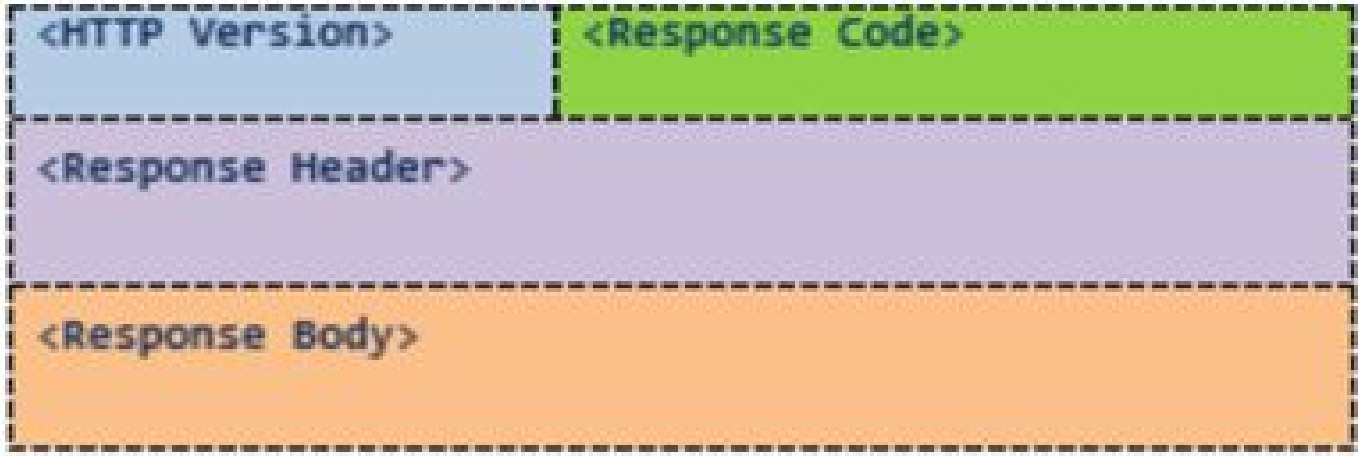
- GET

```
GET http://www.w3.org/Protocols/rfc2616/rfc2616.html HTTP/1.1
Host: www.w3.org
Accept: text/html,application/xhtml+xml,application/xml; …
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 …
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8,hi;q=0.6
```

- POST

```
POST http://MyService/Person/
Host: MyService
Content-Type: text/xml; charset=utf-8
Content-Length: 123
<?xml version="1.0" encoding="utf-8"?>
<Person>
  <ID>1</ID>
  <Name>M Vaqqas</Name>
  <Email>m.vaqqas@gmail.com</Email>
  <Country>India</Country>
</Person>
```

HTTP Response



- <Response Code>: 3-digit HTTP Status code
  - https://en.wikipedia.org/wiki/List_of_HTTP_status_codes
- <Response Body>: Contains Resource Representation

HTTP Response Example

```
1  HTTP/1.1 200 OK
2  Date: Sat, 23 Aug 2014 18:31:04 GMT
3  Server: Apache/2
4  Last-Modified: Wed, 01 Sep 2004 13:24:52 GMT
5  Accept-Ranges: bytes
6  Content-Length: 32859
7  Cache-Control: max-age=21600, must-revalidate
8  Expires: Sun, 24 Aug 2014 00:31:04 GMT
9  Content-Type: text/html; charset=iso-8859-1
10 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR
11 <html xmlns='http://www.w3.org/1999/xhtml'>
12 <head><title>Hypertext Transfer Protocol -- HTTP/1.1</title></head>
13 <body>
14 ...
```

Resource Identifier

- Each resource may have one or more identifiers
- URI:

```
{uri-scheme} ":" {uri-specific-part}
```

- URI examples:
  - `ftp://{host}/{path}`
  - `vnc://{host}`
  - `http://{host}[:{port}]/{path}`
  - `active:fn+operator@uri`

Recommendations for URI

- Plural nouns for sets (collections)
- Singular nouns for individual resources
- Strive for consistent naming
- URI is case-insensitive
- Don’t use verbs to identify a resource
- Include version information

HTTP Verbs

- **Get** is used to retrieve data
- **Post** is used to add data
- **Put** is used to add data or alter existing data
  - *Put should be idempotent which means whether the request is made once or multiple times the effects on the data is exactly the same*
- **Delete** is used to remove data

Python Service Example

```python
@application.route("/api/conversions/temperature/<temp>")
def to_celsius(temp):

    try:
        temptoconvert = float(temp)
    except:
        raise InvalidUsage('Invalid User Input...', status_code=400)

    conv = Converter()
    conv.setTemp(temptoconvert)
    conv.toCelsius()
    celsius = conv.converted_temp
    conv.toFahrenheit()
    fahrenheit = conv.converted_temp

    result = {"input":temptoconvert,
              "celsius":celsius,
              "fahrenheit":fahrenheit}

    return json.dumps(result)
```

```python
@application.errorhandler(InvalidUsage)
def handle_invalid_usage(error):
    response = jsonify(error.to_dict())
    response.status_code = error.status_code
    return response
```

```python
class InvalidUsage(Exception):
    status_code = 400

    def __init__(self, message,
status_code=None, payload=None):
        Exception.__init__(self)
        self.message = message
        if status_code is not None:
            self.status_code =
status_code
        self.payload = payload

    def to_dict(self):
        rv = dict(self.payload or ())
        rv['message'] = self.message
        return rv
```

Exercise: Programming Microservices in the Cloud

- Using the course notes and the Converter sample as a guide, program a RESTful endpoint in your case study project

Chapter Concepts

Microservice Architecture

Programming Microservices

Deploying Microservices with Docker

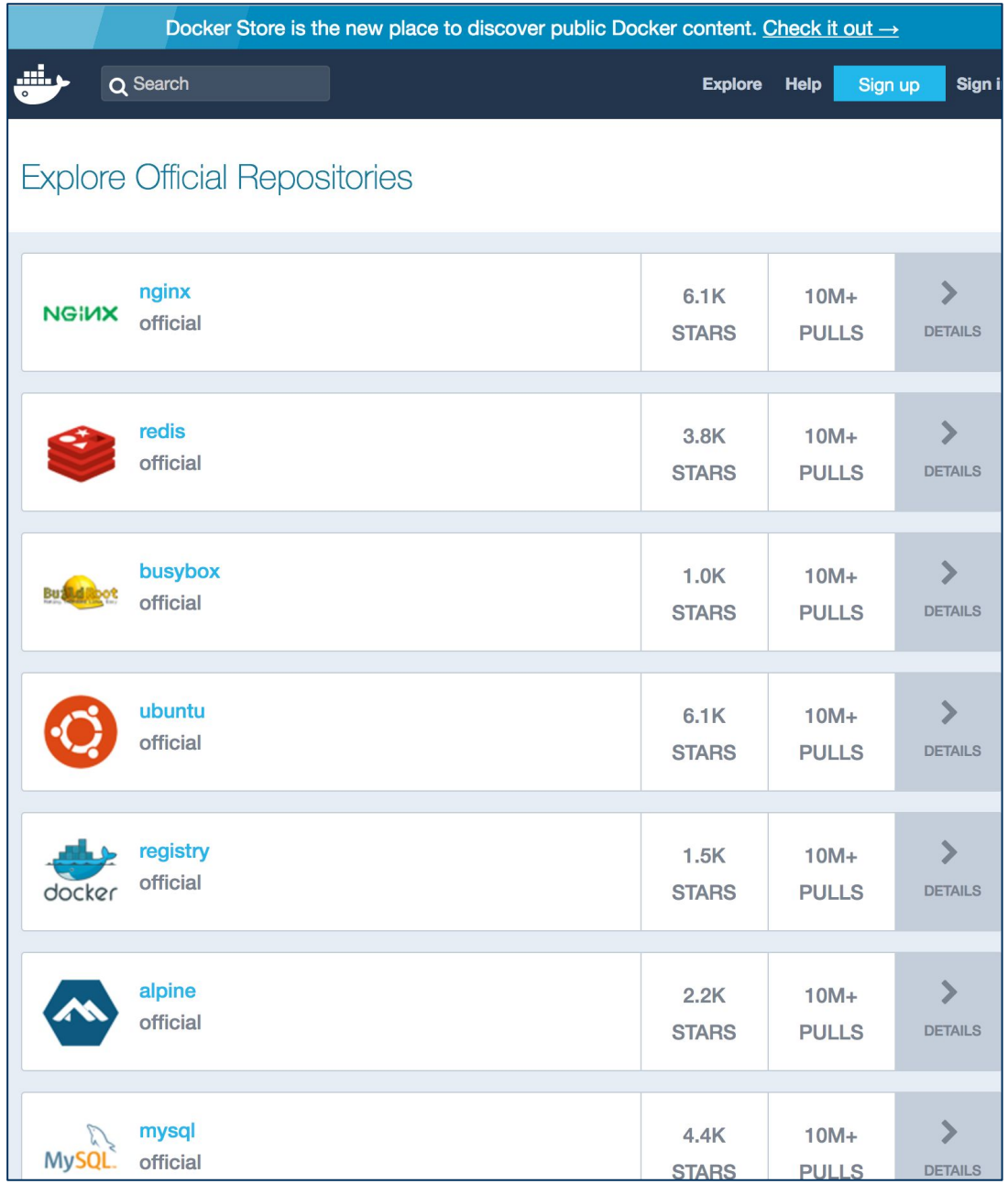
Chapter Summary

Docker

- Docker allows applications or microservices to be deployed to containers
  - http://www.docker.com/
- Multiple containers can run on a single virtual machine
  - Containers allow for even greater optimization of underlying hardware
- Docker images are very light-weight, pre-configured virtual environments
  - Create a Docker image with the required software to run your application
  - Deploy the app to the Docker image
- Docker images will run on any platform
- Docker images allow applications to be easily moved
  - From developer to test to production environments
  - Between local and cloud-based data centers
  - Between different cloud providers
- All major cloud computing vendors support Docker

Docker Hub

- Official repository of Docker images both public and private
- Images for many operating systems and languages
  - Starting points for building your images
- Can upload custom images
  - When deployed onto systems, your custom images are downloaded from the repository
- Can use other Docker repositories, not just Docker Hub



Building Custom Docker Images

- To build a custom image, create a file called Dockerfile
- Steps
  1. Start with a base image from Docker Hub
  2. Identify yourself (*so you can upload your custom image later*)
  3. Install prerequisite software onto the base image
  4. Copy your application onto the image
  5. Configure your application
  6. Start the application

Dockerfile Example for Python App

```
FROM ubuntu:latest
MAINTAINER Joe Smith "jsmith@yahoo.com"
RUN apt-get update -y
RUN apt-get install -y python-pip python-dev build-essential
COPY . /app
WORKDIR /app
RUN pip install -r requirements.txt
ENTRYPOINT ["python"]
CMD ["application.py"]
```

Dockerfile Example for ASP.NET App

```
FROM microsoft/dotnet:latest
MAINTAINER Joe Smith "jsmith@yahoo.com"
COPY ./ /app
WORKDIR /app
RUN ["dotnet", "restore"]
RUN ["dotnet", "build"]
EXPOSE 8080/tcp
ENTRYPOINT ["dotnet", "run", "--server.urls",
"http://0.0.0.0:8080"]
```

Basic Docker Commands

Command	Description
docker build [OPTIONS] PATH | URL | - **Example:** docker build -t drehnstrom/converter-dar:latest .	Build a custom Docker container based on a Dockerfile. Run the command from the same folder as the Dockerfile
docker run [OPTIONS] IMAGE [COMMAND] [ARG...] **Example:** docker run -d -p 8080:8080 drehnstrom/converter-dar	Run a Docker image.
docker ps [OPTIONS	List running docker images. Displays containers and their IDs.
docker stop [OPTIONS] CONTAINER [CONTAINER...] **Example:** docker stop <container-id-here>	Stop a running image.
docker login [OPTIONS] [SERVER]	Login to Docker Hub.
docker push [OPTIONS] NAME[:TAG] **Example:** docker push drehnstrom/converter-dar	Push a container to Docker Hub.

https://docs.docker.com/engine/reference/builder/

Exercise: Getting Started with Docker

- Run the following tutorial to learn the basics of Docker
  https://docs.docker.com/get-started/

Chapter Concepts

Microservice Architecture

Programming Microservices

Deploying Microservices with Docker

Chapter Summary

Chapter Summary

- Microservice architectures divide large applications into a set of small independent services
- Modern web services use a REST architectural style
- REST services take advantage of standards HTTP verbs like GET, POST, PUT, and DELETE
- Docker containers make microservice deployment for efficient and easier

ROITRAINING
MAXIMIZE YOUR TRAINING INVESTMENT

Programming Microservice and Big Data Applications in the Cloud

CHAPTER 4:
CLOUD COMPUTING BIG DATA SERVICES

Chapter Objectives

- Configure and run Hadoop and Spark clusters in the cloud
- Leverage Jupyter and iPython Notebooks for data analysis
- Program data processing pipelines
- Perform big data analysis using no-ops cloud services

Chapter Concepts

Managed Hadoop and Spark

Jupyter

Data Pipelines

Big Data Analysis in the Cloud

Chapter Summary

Hadoop History

- Hadoop was based on a whitepaper describing Google's infrastructure
  - Google File System (GFS) was Google's distributed storage system
  - MapReduce was an algorithm for distributed jobs across many computers
- Yahoo Implemented their version of this system and named it Hadoop
  - Yahoo later released the Hadoop project as an open-source project
  - Currently managed by the Apache Software Foundation

Apache Hadoop Ecosystem

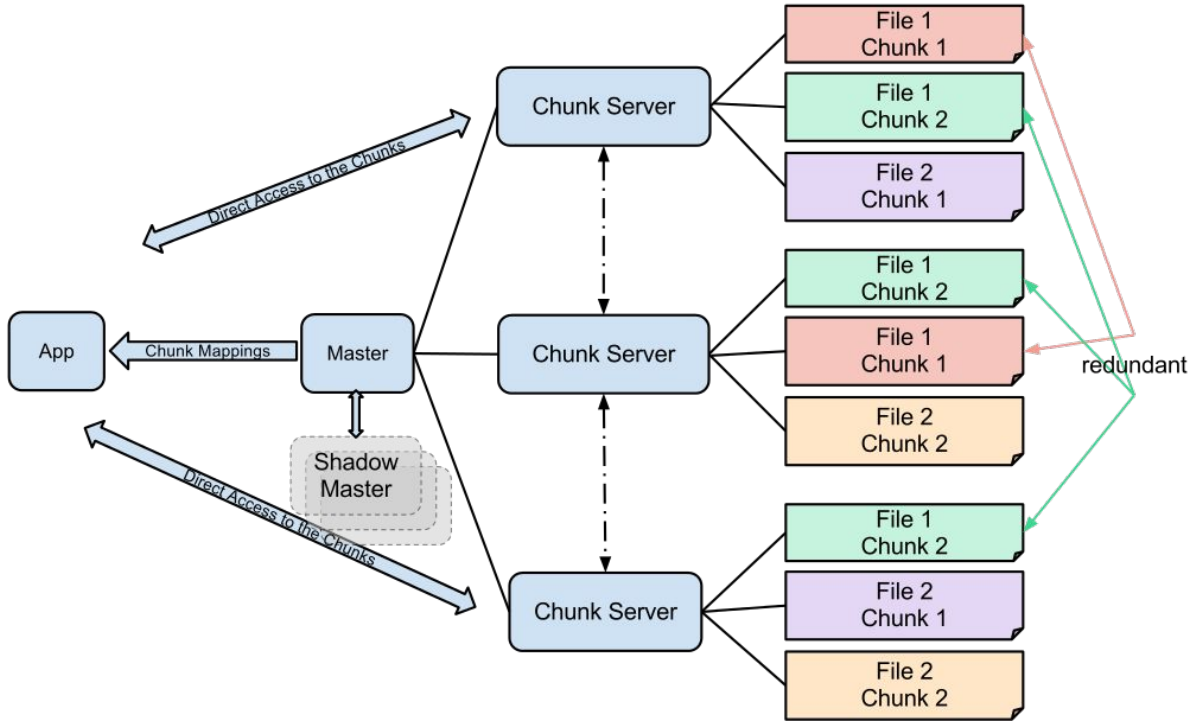
- Hadoop consists of:
  - Hadoop Distributed File System (HDFS) for storage
  - Hadoop MapReduce distributed computing system
  - Hadoop YARN for job scheduling and cluster management
- Many additional projects are closely associated with Hadoop
  - Pig
  - Hive
  - Spark
  - PySpark
  - Etc.

MapReduce

- MapReduce is a way of dividing a large task onto multiple machines to get it done faster
  - Map step sorts and filters the data
  - Reduce step summarizes the results
- Imagine you have a swimming pool filled with blue, red, yellow, and green marbles, and you want to know how many of each there are, and you want to know quick
  - Get 1000 people to sort the marbles into separate boxes (Map)
  - Count the marbles in each box (Reduce)

HDFS

- To process huge amounts of data, you need many drives working in parallel
  - Large files are split into pieces
  - Many computers can read and write to many disks for speed
  - Each piece is written to multiple disks for redundancy
  - A master computer keeps track of all the files



Apache Pig

- Platform for building parallel processing jobs on Hadoop
- Uses a language called Pig Latin
  - Simpler than using Java to program MapReduce jobs
- See https://pig.apache.org/

```
input_lines = LOAD '/tmp/my-data' AS (line:chararray);
words = FOREACH input_lines GENERATE FLATTEN(TOKENIZE(line)) AS word;
filtered_words = FILTER words BY word MATCHES '\\w+';
word_groups = GROUP filtered_words BY word;
word_count = FOREACH word_groups GENERATE COUNT(filtered_words) AS count, group AS word;
```

Apache Hive

- Provides a SQL-like language for running jobs on Hadoop clusters

```
DROP TABLE IF EXISTS docs;
CREATE TABLE docs (line STRING);
LOAD DATA INPATH 'input_file' OVERWRITE INTO TABLE docs;
CREATE TABLE word_counts AS
SELECT word, count(1) AS count FROM
  (SELECT explode(split(line, '\s')) AS word FROM docs) temp
  GROUP BY word
  ORDER BY word;
```

Apache Spark

- Spark provides an in-memory dataset that enables much faster processing than traditional MapReduce
- Provides programming interfaces in multiple languages
  - Python, Java, Scala, R, etc.
- Can read data from multiple sources including HDFS, but also Amazon S3 and Google Cloud Storage

PySpark

- Provides a Python interface for programming Spark jobs

```
text_file = spark.textFile("hdfs://...")

text_file.flatMap(lambda line: line.split())
    .map(lambda word: (word, 1))
    .reduceByKey(lambda a, b: a+b)
```

Apache Spark SQL

- Provides an interface for querying Spark data sets using SQL

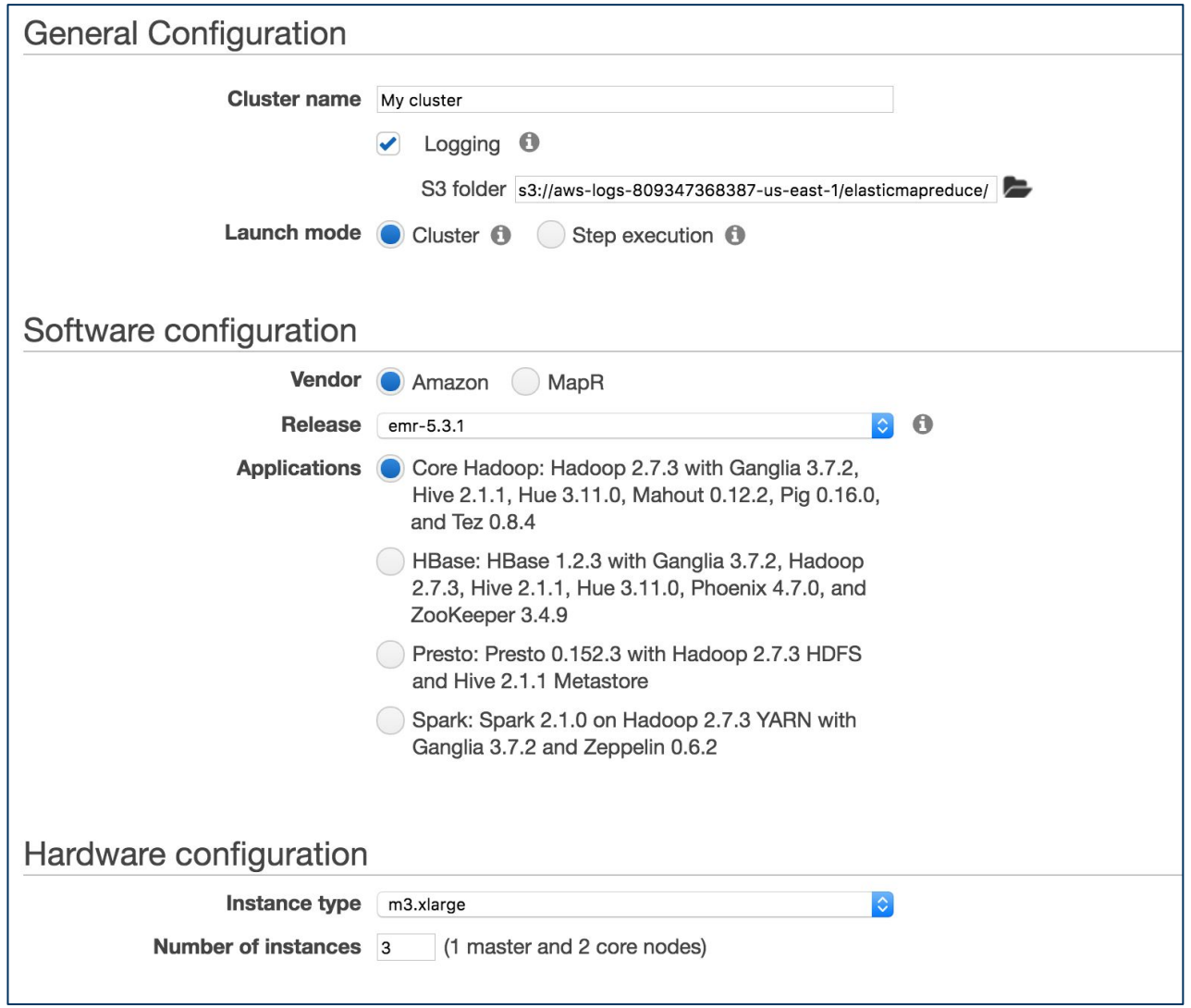
```
context = HiveContext(sc)
results = context.sql(
  "SELECT * FROM people")
names = results.map(lambda p: p.name)
```

```
context.jsonFile("s3n://...")
  .registerTempTable("json")
results = context.sql(
  """SELECT *
     FROM people
     JOIN json ..."""")
```

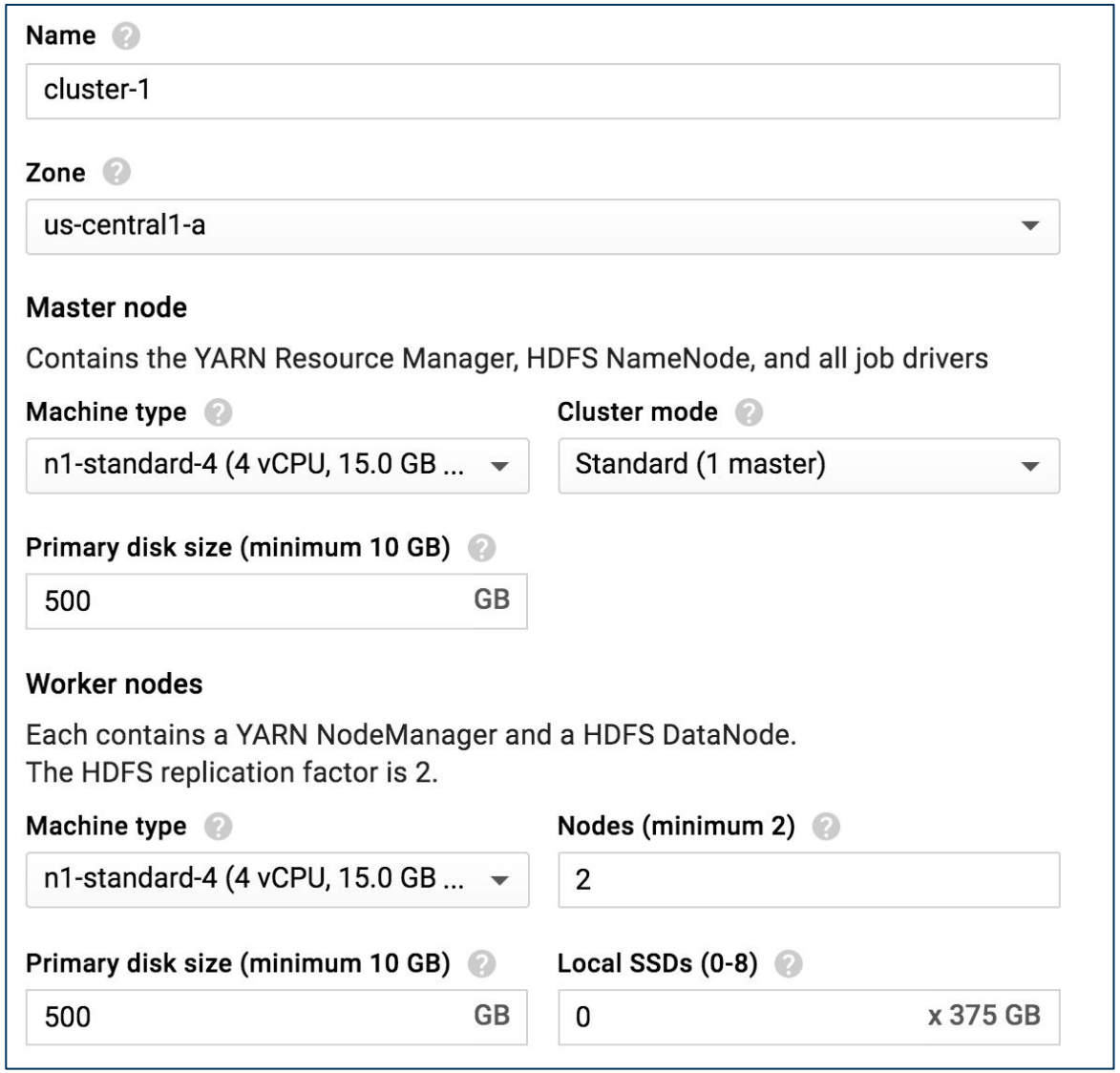
Hadoop in the Cloud

- Cloud is ideal for running Hadoop and Spark jobs
  - Little or no administration
  - Can create clusters in minutes
  - Destroy clusters as soon as jobs are completed
  - Can create clusters of any size
- Integrate with cloud-based storage systems
  - S3 on AWS and Google Cloud Storage
  - BigTable on GCP and DynamoDB on AWS

AWS Elastic MapReduce (EMR)



Google Cloud Dataproc



Exercise: Creating Hadoop and Spark Clusters

- AWS Elastic MapReduce
  https://storage.googleapis.com/roi-code-labs/codelabs/labs/793b-aws-emr-cluster/index.html

- GCP Dataproc
  https://storage.googleapis.com/roi-code-labs/codelabs/labs/793b-creating-dataproc-clusters/index.html

Chapter Concepts

Managed Hadoop and Spark

Jupyter

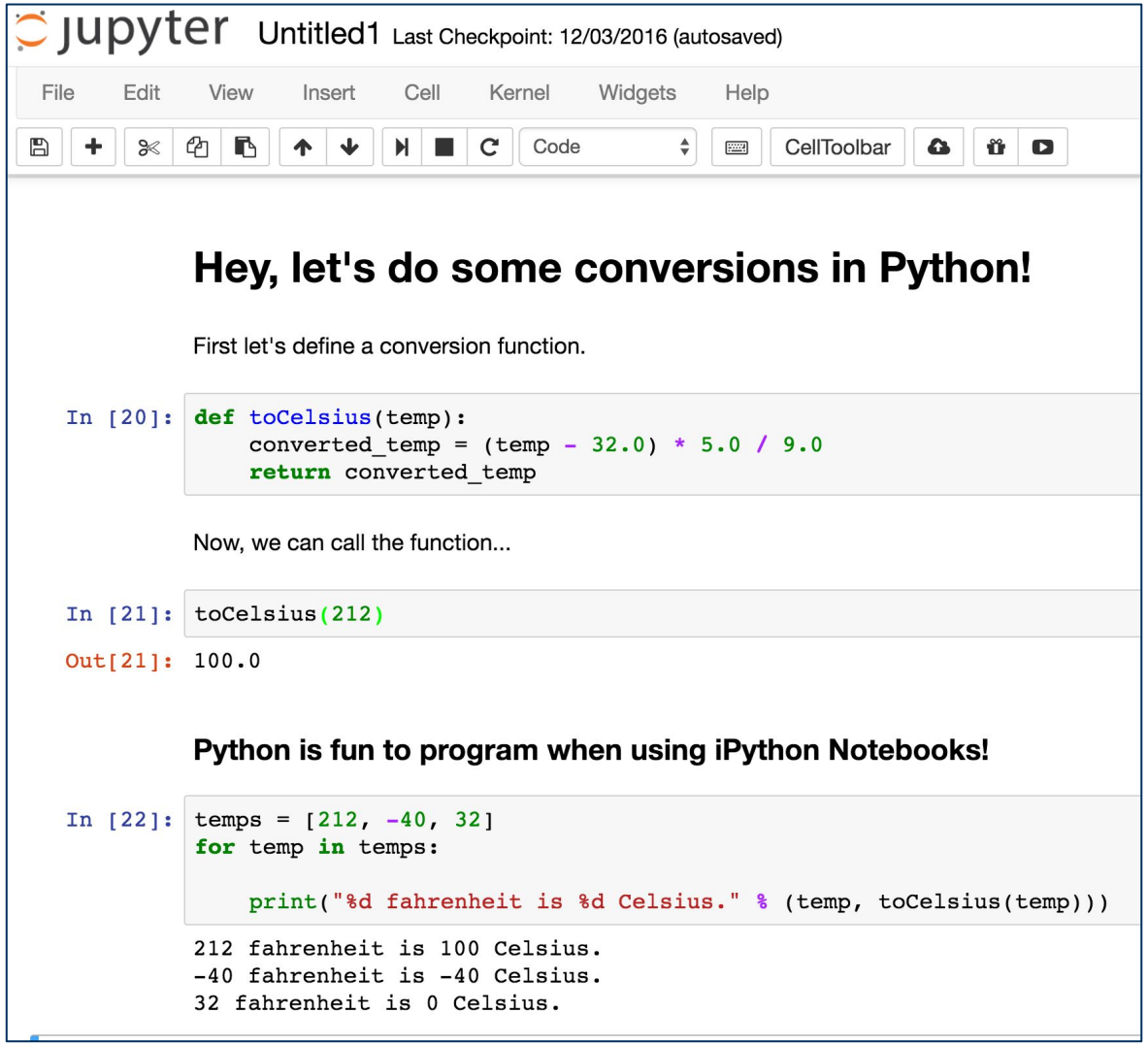
Data Pipelines

Big Data Analysis in the Cloud

Chapter Summary

iPython Notebooks

- Combine markdown and code to create interactive applications
- Support for many languages
- Can edit code and rerun interactively
- Very useful for data analysis and machine learning



Jupyter

- Is a web-based runtime for executing iPython notebooks
- Can be installed on any machine
  - Developers can run Jupyter on their local machines
  - Jupyter can be run a servers in the cloud
  - Jupyter can be installed on a Hadoop cluster to run Spark jobs interactively
- Just type `jupyter notebook` in in a terminal window to start

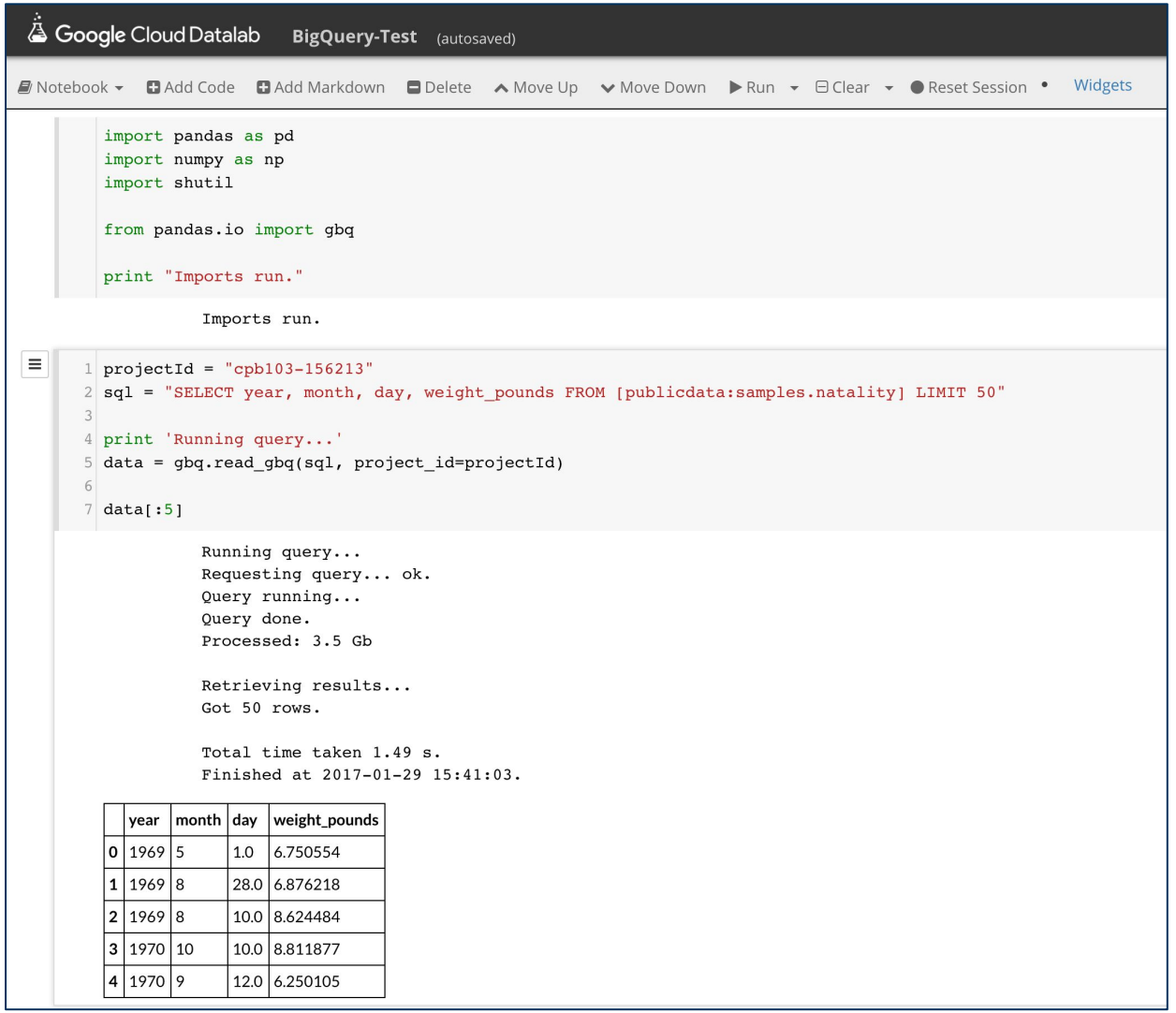
Running Jupyter on EMR Clusters

- Install Jupyter on EMR using a bootstrap action
- Jupyter will run on the port specified in optional arguments

```
aws emr create-cluster --release-label emr-5.2.0 \
  --name 'my-jupyter-cluster' \
  --applications Name=Hadoop Name=Hive Name=Spark Name=Pig Name=Tez Name=Ganglia Name=Presto \
  --ec2-attributes KeyName=<my-key-pair>,InstanceProfile=EMR_EC2_DefaultRole \
  --service-role EMR_DefaultRole \
  --instance-groups \
    InstanceGroupType=MASTER,InstanceCount=1,InstanceType=c3.xlarge \
    InstanceGroupType=CORE,InstanceCount=2,InstanceType=c3.xlarge \
  --region us-east-1 \
  --log-uri s3://<my-bucket-name>/emr-logs/ \
  --bootstrap-actions \
    Name='Install Jupyter
notebook',Path="s3://aws-bigdata-blog/artifacts/aws-blog-emr-jupyter/install-jupyter-emr5.sh",Args=
[--r,--julia,--toree,--torch,--ruby,--ds-packages,--ml-packages,--python-packages,'ggplot
nilearn',--port,8880,--password,jupyter,--jupyterhub,--jupyterhub-port,8001,--cached-install,
--notebook-dir,s3://<my-bucket-name>/notebooks/,--copy-samples]
```

Google Cloud Datalab

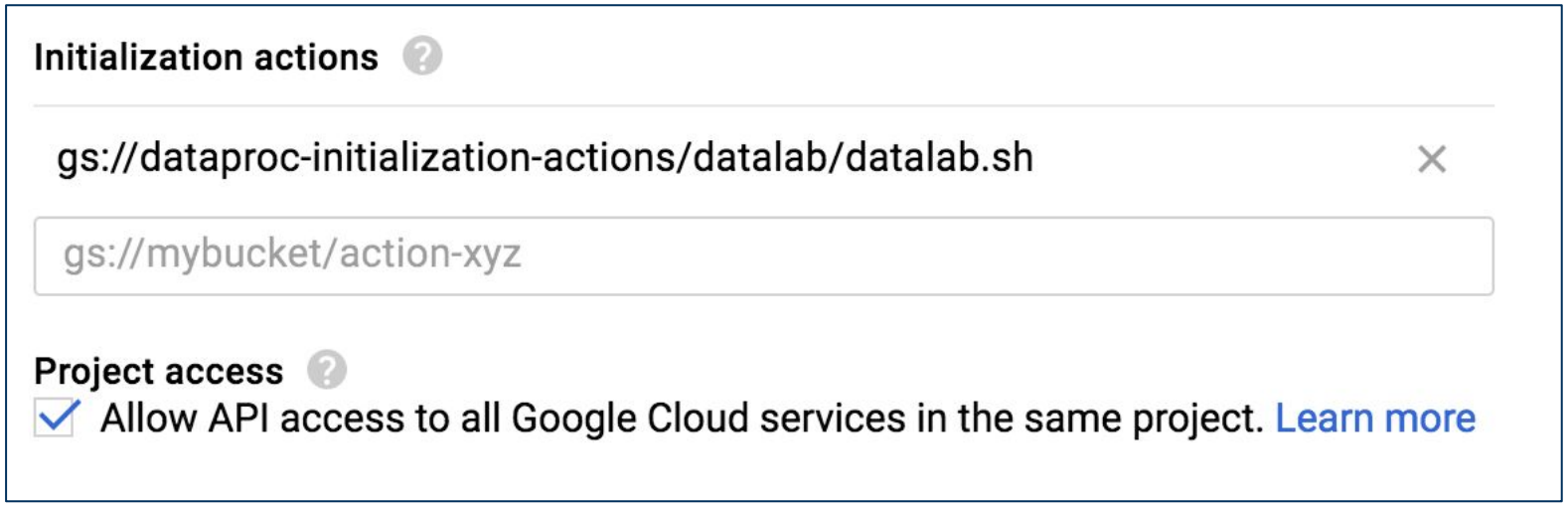
- Customized version of Jupyter optimized for Google services



	year	month	day	weight_pounds
0	1969	5	1.0	6.750554
1	1969	8	28.0	6.876218
2	1969	8	10.0	8.624484
3	1970	10	10.0	8.811877
4	1970	9	12.0	6.250105

Running Datalab on Dataproc Clusters

- Use initialization actions to install Google Cloud Datalab
  - Can alternatively install Jupyter
- For additional Initialization actions see:
  - https://cloud.google.com/dataproc/docs/concepts/init-actions
  - https://github.com/GoogleCloudPlatform/dataproc-initialization-actions



Exercise: Programming Jupyter Notebooks

- AWS Programming Jupyter Notebooks
  https://storage.googleapis.com/roi-code-labs/codelabs/labs/793b-aws-emr-jupyter/index.html

- GCP Programming Jupyter Notebooks
  https://storage.googleapis.com/roi-code-labs/codelabs/labs/793b-gcp-programming-jupyter-notebooks/index.html

Chapter Concepts

Managed Hadoop and Spark

Jupyter

Data Pipelines

Big Data Analysis in the Cloud

Chapter Summary

Data Pipelines

- Perform a set of actions in a chain
  - Data from one or more sources is read into the pipeline
  - Actions are performed on the data to manipulate or transform it
  - The manipulated results are sent as output from the pipeline
- Actions within a data flow can run at the same time (concurrently)
- MapReduce jobs are examples of data flows at scale
  - Multiple nodes read data from disk and perform an initial map step
  - Data is then organized by key (the shuffle step)
  - Keyed data is processed separately in parallel (the reduce step)

Batch vs. Streaming Data Flows

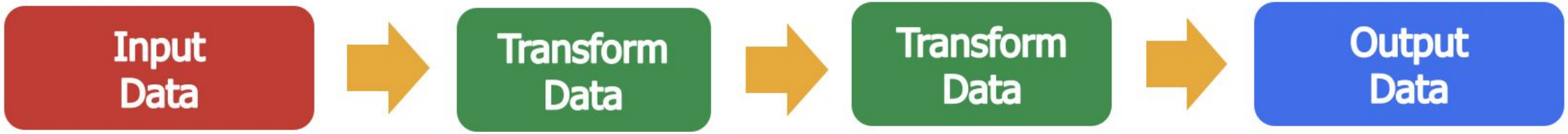
- Batch data flows process big chunks of data at set intervals
  - Analyzing daily logs
  - Importing monthly sales
  - Periodic data conversions
- Streaming data flows process data as it is accumulated
  - Analyzing traffic to determine the quickest route
  - What tweets are trending right now
  - What products are selling today

Moving Data

- The simplest data flow is simply moving data from one location to another
  - Hardly a difficult problem unless huge amounts of data are involved

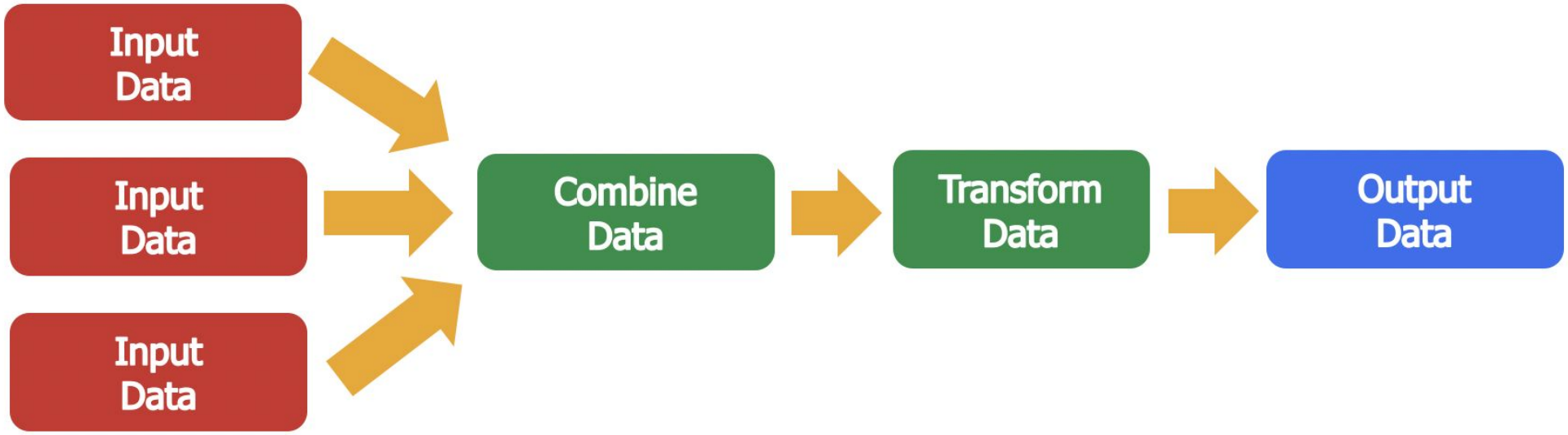
Moving and Transforming Data

- Data is moved from one place to another, but altered along the way
  - More complicated as the amount of data increases and the transformations become more complex
  - More data and more processing means more time
- More machines can participate in the process to get the job done faster
  - This increases cost and complexity



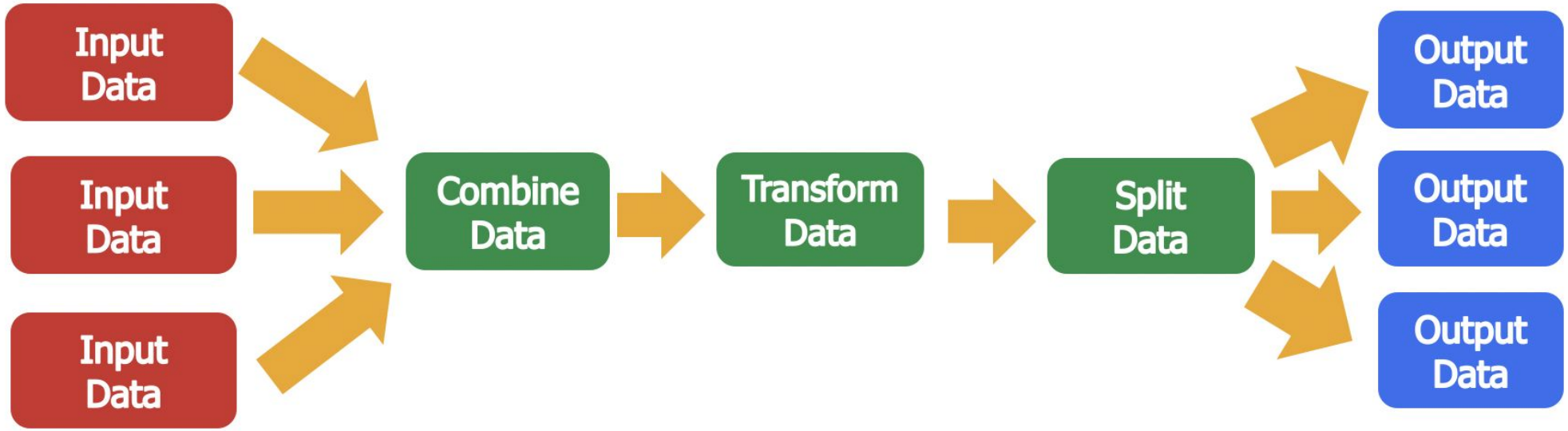
Multiple Inputs

- Sometimes a data flow requires input from multiple data sources
  - Data from a web log is combined with sales data from a database



Multiple Outputs

- Sometimes a data flow needs to output the data more than one way
  - Storage is cheap relative to processing power and time
  - Outputting the same data in multiple, different formats can make data analysis easier and more efficient



Programming Pipelines with Apache Beam

- Apache Beam is an open-source framework for programming pipelines
  - Supports both batch and streaming pipelines
  - Runs on multiple environments including Apache Spark and Google Cloud Dataflow
  - Supports coding in both Java and Python
- Can connect to many data sources
  - Amazon Kinesis, FileInputFormat in HDFS, Apache HBase, Apache Kafka, Avro Files, Google BigQuery, Google Cloud Bigtable, Google Cloud Datastore, Google Cloud Pub/Sub, Google Cloud Storage, JDBC, Java Message Service (JMS), MongoDB, Text Files, XML Files
- For more information see: https://beam.apache.org/

Word Count Example (Java)

```
p.apply(TextIO.Read.from("gs://apache-beam-samples/shakespeare/*"))
  .apply("ExtractWords", ParDo.of(new DoFn<String, String>() {
      @ProcessElement
      public void processElement(ProcessContext c) {
          for (String word : c.element().split("[^a-zA-Z']+")) {
              if (!word.isEmpty()) {
                  c.output(word);
              }
          }
      }
  }))
  .apply(Count.<String>perElement())
  .apply("FormatResults", MapElements.via(new SimpleFunction<KV<String, Long>, String>() {
      @Override
      public String apply(KV<String, Long> input) {
          return input.getKey() + ": " + input.getValue();
      }
  }))
  .apply(TextIO.Write.to("wordcounts"));
```

Word Count Example (Python)

```
p = beam.Pipeline(options=options)

p | beam.io.ReadFromText('gs://dataflow-samples/shakespeare/kinglear.txt')
  | 'ExtractWords' >> beam.FlatMap(lambda x: re.findall(r'[A-Za-z\']+', x))
  | beam.combiners.Count.PerElement()
  | beam.Map(lambda (word, count): '%s: %s' % (word, count))
  | beam.io.WriteToText('gs://my-bucket/counts.txt')

result = p.run()
```

Chapter Concepts

Managed Hadoop and Spark

Jupyter

Data Pipelines

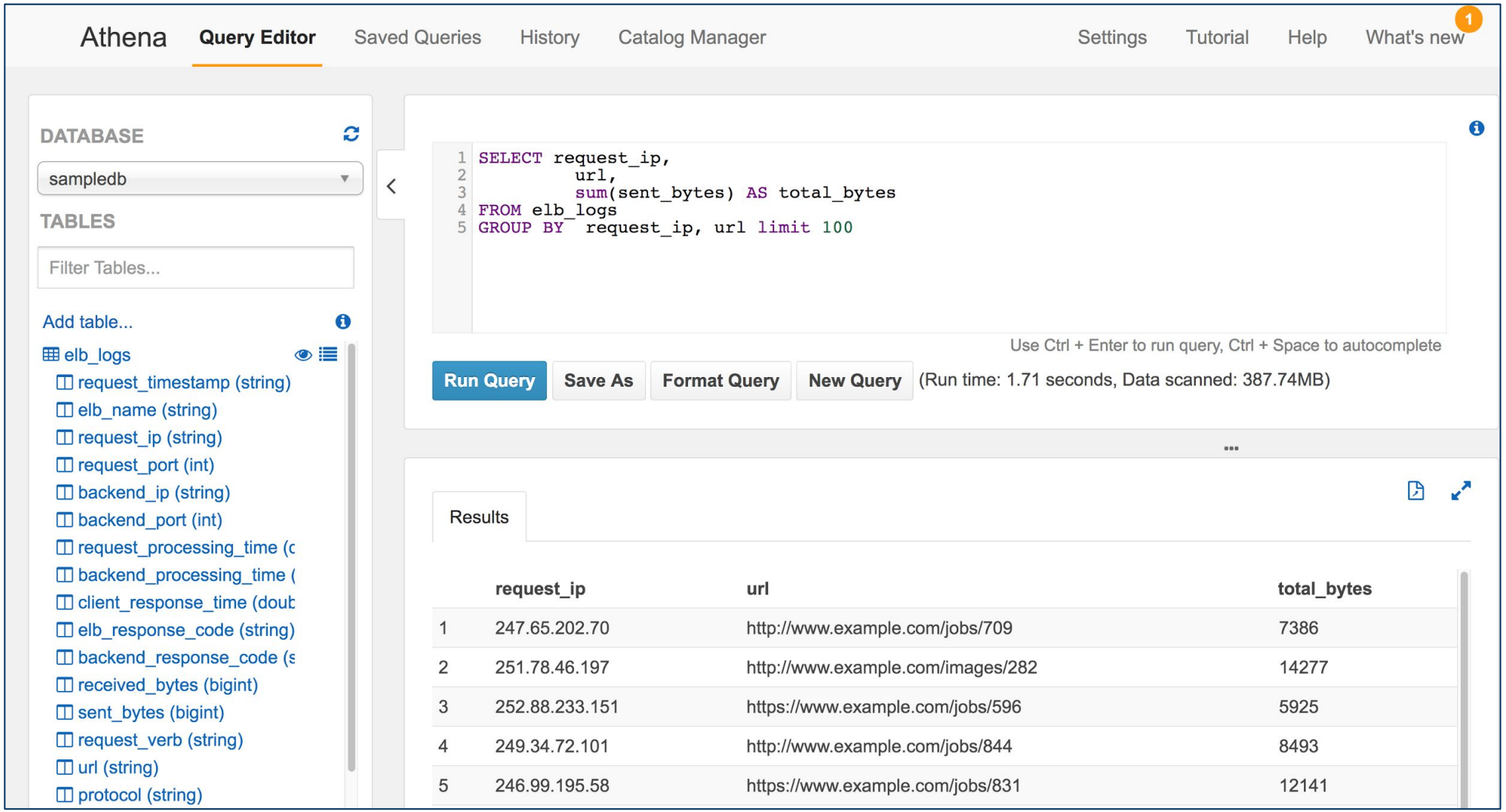
Big Data Analysis in the Cloud

Chapter Summary

AWS Athena

- Interactive query service that uses standard SQL to query data
- Query data that is stored in S3
  - Supports CSV, JSON, and other formats
- Serverless infrastructure requires no setup or administration
  - Uses Presto query engine
  - https://prestodb.io/
- Inexpensive
  - Data is simply stored in S3
  - Cost for querying data is $5/TB of data processed
- For more information see: https://aws.amazon.com/documentation/athena/

Athena Illustrated



Google BigQuery

- Massively scalable no-ops data warehouse and analysis service
- Extremely cheap storage solution
  - 2 cents per GB for first 3 months, then 1 cent per GB after that
- Extremely fast and inexpensive querying engine
  - $5 per TB of data processed
  - Can query petabytes of data
- Based in Google's internal Dremel Querying engine used inside Google for many years

BigQuery Illustrated

ThisYearLastYearSalesByVendor-With If statements

Query Editor | UDF Editor

SQL

```
1  SELECT
2    b.vendor AS Vendor,
3    ROUND(SUM(IF(a.yeardate = '2014', a.total, 0)),2) AS LYSales,
4    ROUND(SUM(IF(a.yeardate = '2015', a.total, 0)),2) AS TYSales
5  FROM (
6    SELECT
7      SUBSTR(date,0,4) AS yeardate,
8      vendor_no,
9      total
10   FROM
11     [cpb200 liquor sales.sales]
```

Ctrl + Enter: run query, Tab or Ctrl + Space: autocomplete.

RUN QUERY | Save Query | Save View | Format Query | Show Options

Query complete (2.0s elapsed, 72.0 MB processed)

Results | Explanation | Job Information | Download as CSV | Download as JSON | Save as Table | Save to Google Sheets

Row	Vendor	LYSales	TYSales
1	35 Maple Street	41895.56	6830.02
2	A Hardy / U.S.A., Ltd.	1211683.68	206.55
3	ASDSpirits, LLC	10408.45	3860.4
4	Adamba Imports Int'l Inc.	107.88	0.0
5	Anchor Distilling (PREISS IMPORTS)	62161.27	6072.78

Table | JSON

First < Prev Rows 1 - 5 of 25 Next > Last

Exercise: Big Data Analysis

- AWS Athena
  https://storage.googleapis.com/roi-code-labs/codelabs/labs/793b-aws-athena/index.html

- GCP BigQuery
  https://storage.googleapis.com/roi-code-labs/codelabs/labs/793b-gcp-bigquery/index.html

Chapter Concepts

Managed Hadoop and Spark

Jupyter

Data Pipelines

Big Data Analysis in the Cloud

Chapter Summary

Chapter Summary

- Hadoop and Spark clusters can be easily and quickly provisioned in the cloud using AWS Elastic MapReduce or Google Cloud Dataproc
- Jupyter notebooks are an interactive way of doing data analysis in Python, Spark, and other languages
- Data pipelines can be written using Apache Beam and executed on multiple platforms including Spark clusters and Google Cloud Dataflow
- AWS Athena and Google BigQuery are cost-effective no-ops data analysis services

ROITRAINING
MAXIMIZE YOUR TRAINING INVESTMENT

Programming Microservice and Big Data Applications in the Cloud

CHAPTER 5: REAL-TIME DATA ANALYSIS

Chapter Objectives

In this chapter, we will:

- Analyze data in real time
- Develop asynchronous messaging applications with Google Pub/Sub
- Build real-time data analysis pipelines with Amazon Kinesis

Chapter Concepts

Real-Time Messaging

Google Pub/Sub

AWS Kinesis

Chapter Summary

Real-Time Analytics

- It's not good enough today to run the yesterday's sales report
  - Users want to see current data right now
- Real-time asynchronous messaging applications enable this behavior
  - Web apps, phones, IoT devices, and so on send data to the cloud
  - This data is collected and processed
  - Processed data is delivered to some analytics platform
- On GCP, Pub/Sub is used for messaging and data is sent to BigQuery
- On AWS, Kinesis collects the data and results are sent to S3 or Redshift
  - Data sent to S3 can be analyzed using Athena

Chapter Concepts

Real-Time Messaging

Google Pub/Sub

AWS Kinesis

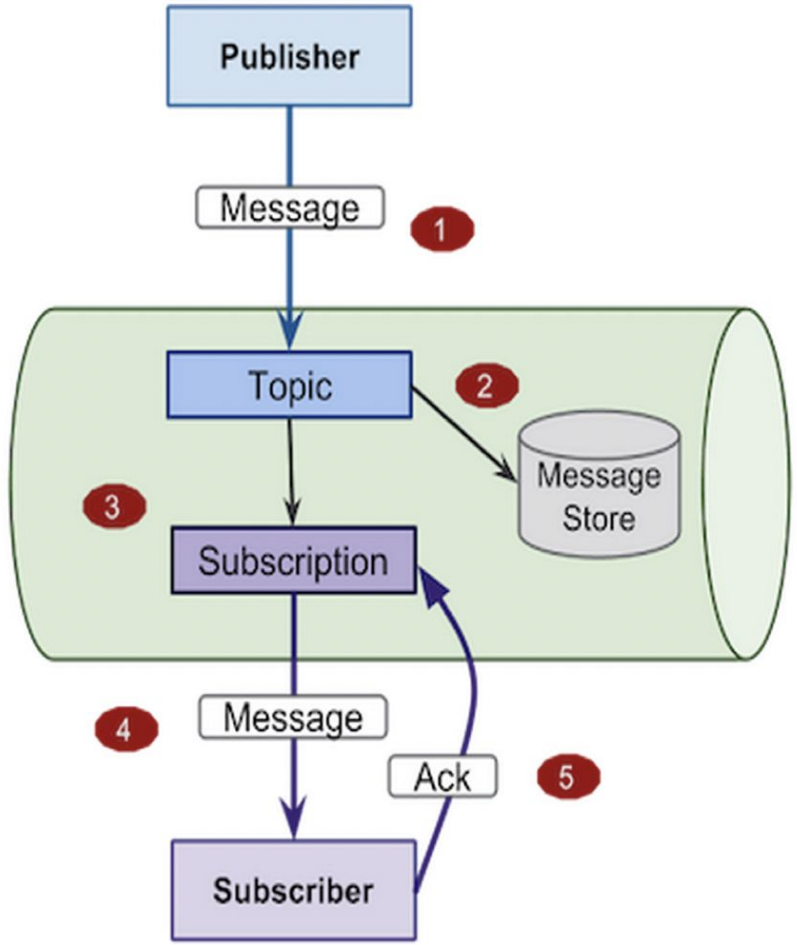
Chapter Summary

Pub/Sub Overview

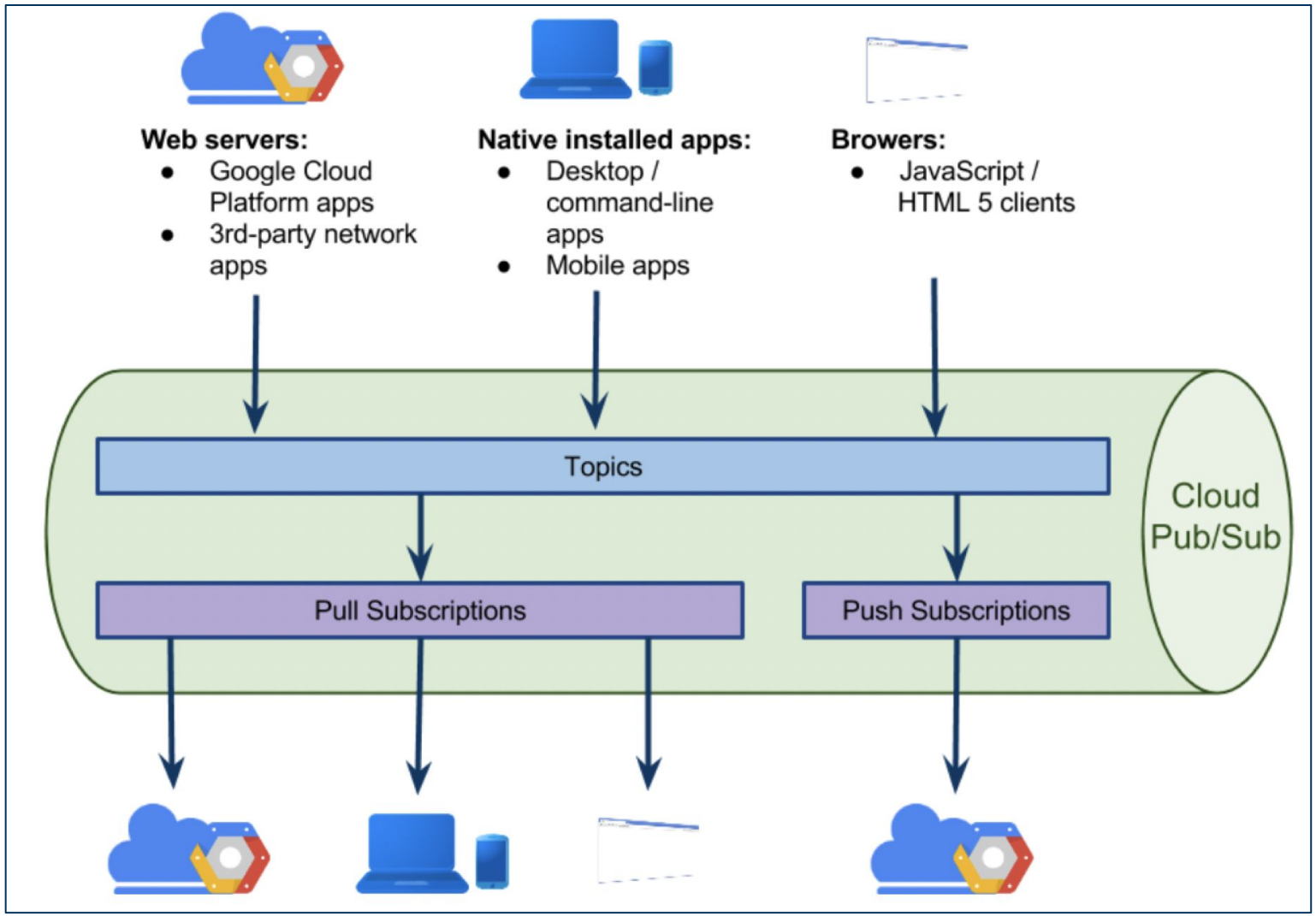
- Pub/Sub is a fully managed, massively scalable messaging service
  - It allows messages to be sent between independent applications
  - Pub/Sub can scale to millions of messages per second
- Pub/Sub messages can be sent and received via HTTPS
  - Any device, computer, or application can use it
  - Pub/Sub supports multiple senders and receivers simultaneously
- Pub/Sub is a global service
  - Messages are copied to multiple zones for greater fault tolerance
  - Uses dedicated resources in every region for fast delivery worldwide
- Pub/Sub is secure, all message are encrypted at rest and in transit

Topics and Subscriptions

- Messages in Pub/Sub are sent to a Topic
  - Messages contain data
- Topics are endpoints where messages are sent
- Subscriptions represent a stream of messages within a topic
  - Topics can contain multiple subscriptions
  - Each subscription belongs to one topic
- Subscribers get messages from subscriptions

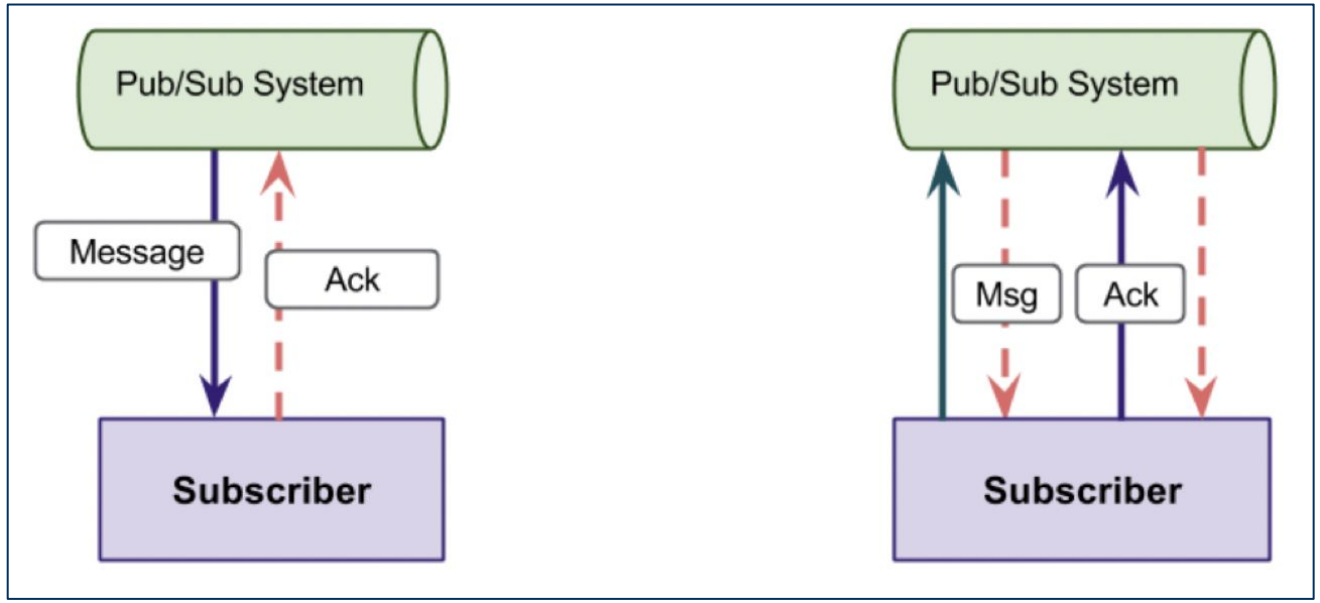


Pub/Sub Integrates with Any Application



Push and Pull Subscriptions

- Subscriptions have a delivery mode of either a push or pull
  - Push messages are automatically sent to the subscriber via an endpoint
  - Acknowledgement of the message is implied by a response code 200
- Pull messages must be requested by the subscriber
  - Subscriber calls the pull() method of the Pub/Sub API
  - If a message exists, it is sent
  - Subscriber then calls the acknowledge() method



Processing Pub/Sub Messages

- Subscribers can be AppEngine apps, Dataflow apps, or something else
- Process messages in real time and send them to BigQuery
- BigQuery can then be used for live analysis

Chapter Concepts

Real-Time Messaging

Google Pub/Sub

AWS Kinesis

Chapter Summary

Amazon Kinesis

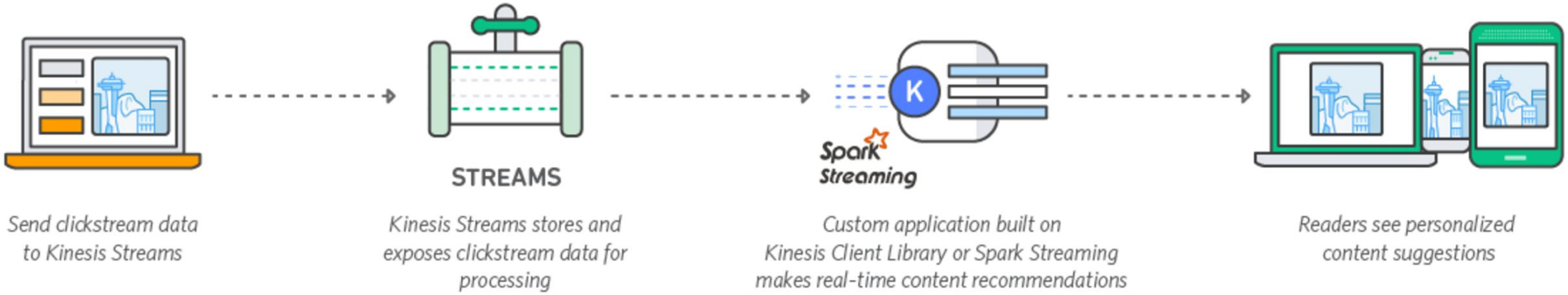
- Kinesis is a set of services that enable real-time data processing and analytics in AWS
  - Can stream huge amounts of data into various AWS data services
  - Can create custom streams
  - Analytics service allows for easy data processing with SQL

Kinesis Firehose

- Steam data into Kinesis Analytics, S3, RedShift, or DynamoDB
- Process the data with your preferred tools
- Autoscales to meet demand
- Pay for only the amount of data streamed through the service

Kinesis Streams

- Provides the ability to create custom streams for applications
- Configure how much resources to allocate to a stream based on capacity required



Kinesis Analytics

- Connect to real-time data sources like Kinesis Firehose
- Process data with standard SQL
- Output processed results to preferred destination like S3 or Redshift

Capture streaming data with Kinesis Streams or Kinesis Firehose

Run standard SQL queries against data streams

Kinesis Analytics can send processed data to analytics tools so you can create alerts and respond in real-time

Chapter Concepts

Asynchronous Messaging

Google Pub/Sub

AWS Kinesis

Chapter Summary

Chapter Summary

- Google Pub/Sub provides massively scalable, secure and reliable asynchronous messaging
- Amazon Kinesis is a set of services for building real-time data analysis applications

ROITRAINING

MAXIMIZE YOUR TRAINING INVESTMENT

Programming Microservice and Big Data Applications in the Cloud

CHAPTER 6: MACHINE LEARNING

Chapter Objectives

In this chapter, we will:

- Define machine learning and its basic concepts
- Train machine learning models using cloud machine learning services
- Add advanced, pre-built machine learning models like image recognition, speech, translations, and language to our applications

Chapter Concepts

Machine Learning Basics

Building Machine Learning Models

Leveraging Existing Machine Learning Models

Chapter Summary

What Is Machine Learning?

Machine learning

From Wikipedia, the free encyclopedia

For the journal, see Machine Learning (journal).

Machine learning is the subfield of computer science that gives computers the ability to learn without being explicitly programmed (Arthur Samuel, 1959).[1] Evolved from the study of pattern recognition and computational learning theory in artificial intelligence,[2] machine learning explores the study and construction of algorithms that can learn from and make predictions on data[3] – such algorithms overcome following strictly static program instructions by making data driven predictions or decisions,[4]:2 through building a model from sample inputs. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms is infeasible; example applications include spam filtering, detection of network intruders or malicious insiders working towards a data breach,[5] optical character recognition (OCR),[6] search engines and computer vision.

Examples of Machine Learning

- Recommendation engines
- Customer churn analysis
- Fraud detection
- Image recognition
- Sentiment analysis
- Translation
- Voice recognition
- Many more…

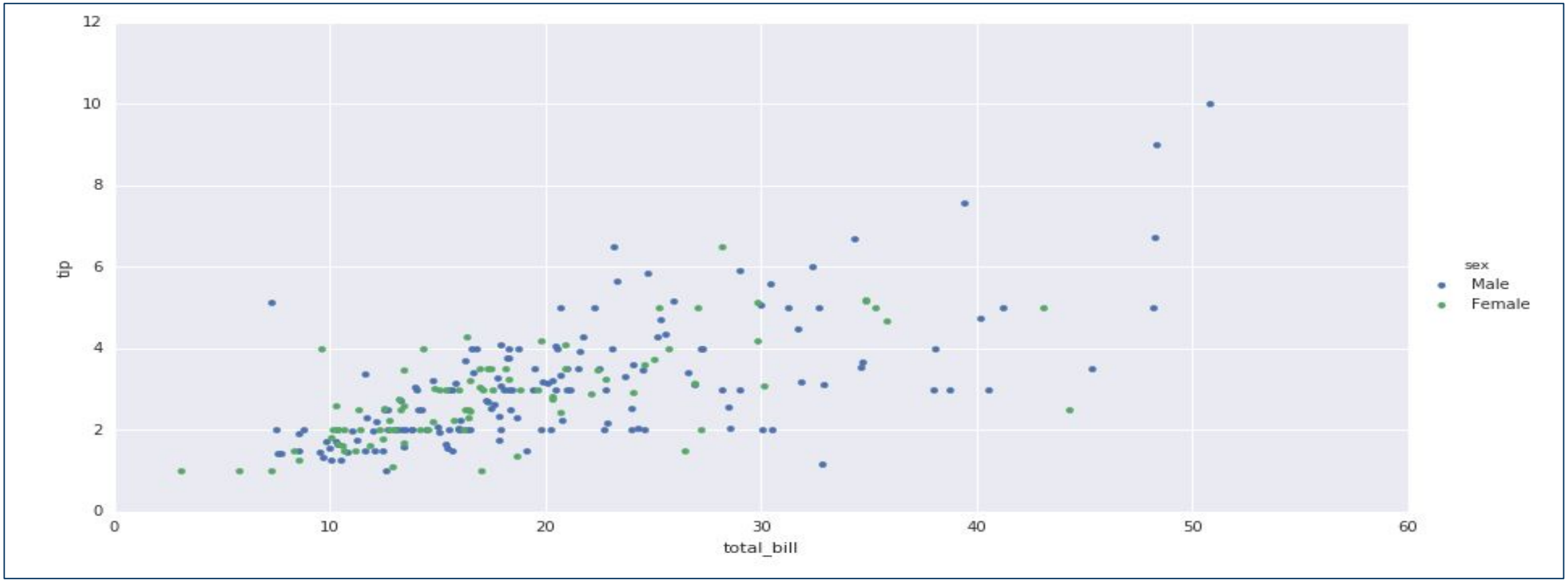
When to Use Machine Learning

- Use machine learning when it would be too hard to program something with traditional means
  - Too many variables to solve a problem with if-statements

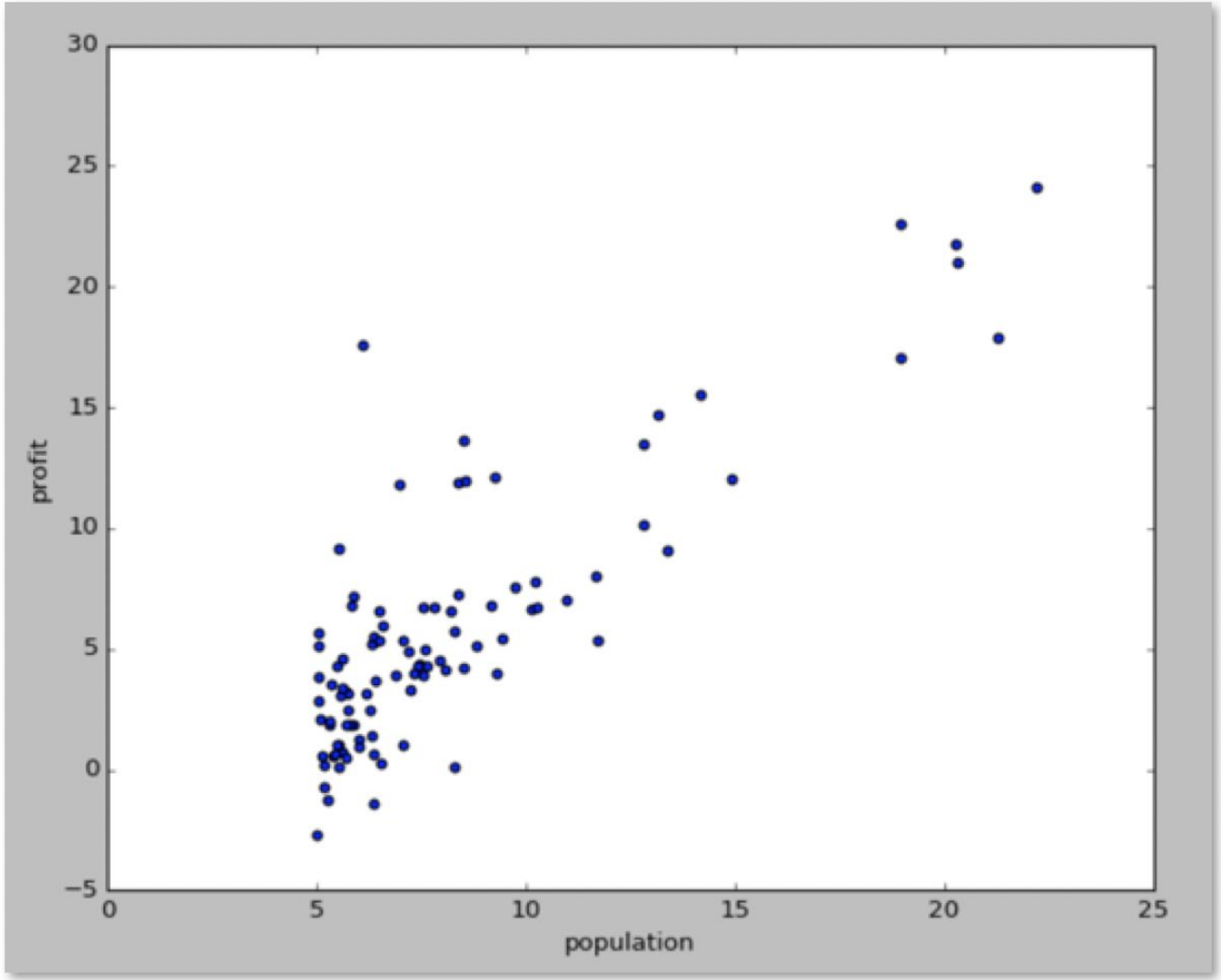
Types of Machine Learning

Regression
Predict the tip amount

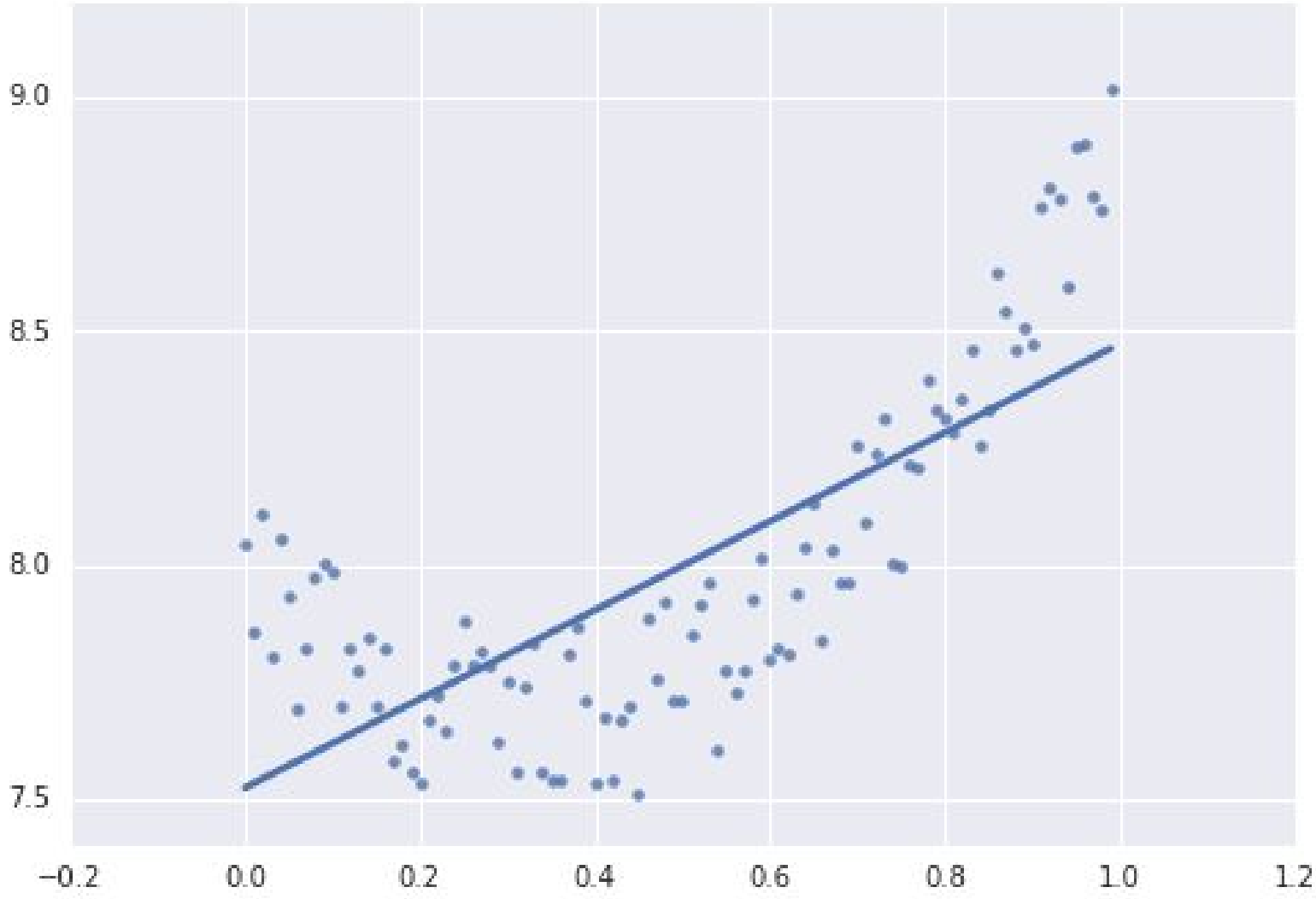
Classification
Predict the gender of the customer



Start with Some Data



Fit a Line



Hypothesis Function

- Straight line is a good first approximation for the data
  - The line has the following equation:

    y = p0 + p1 * x
  - p0 and p1 are the parameters of the equation
  - x is the input value
  - y is the target value
- Need to determine the best values of p0 and p1
- Requires a cost function

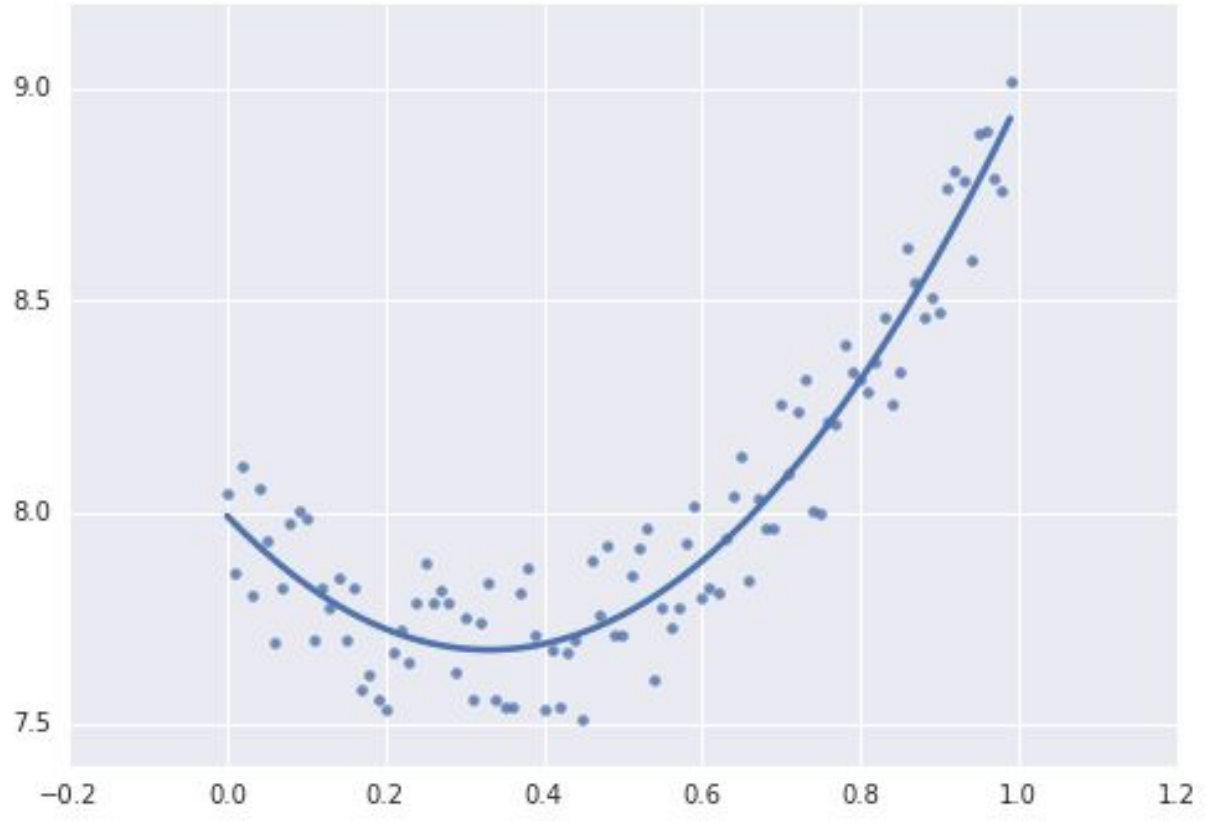
Cost Functions

- Calculate how much error is in the hypothesis function
- Common to use the mean squared error
- Changing the hypothesis function will result in a change in the error
  - Goal is to minimize the error

Actual	Predicted	Error	Error Squared
10	8	-2	4
12	16	4	16
14	13	-1	1
22	25	3	9
		Mean Squared Error:	7.5

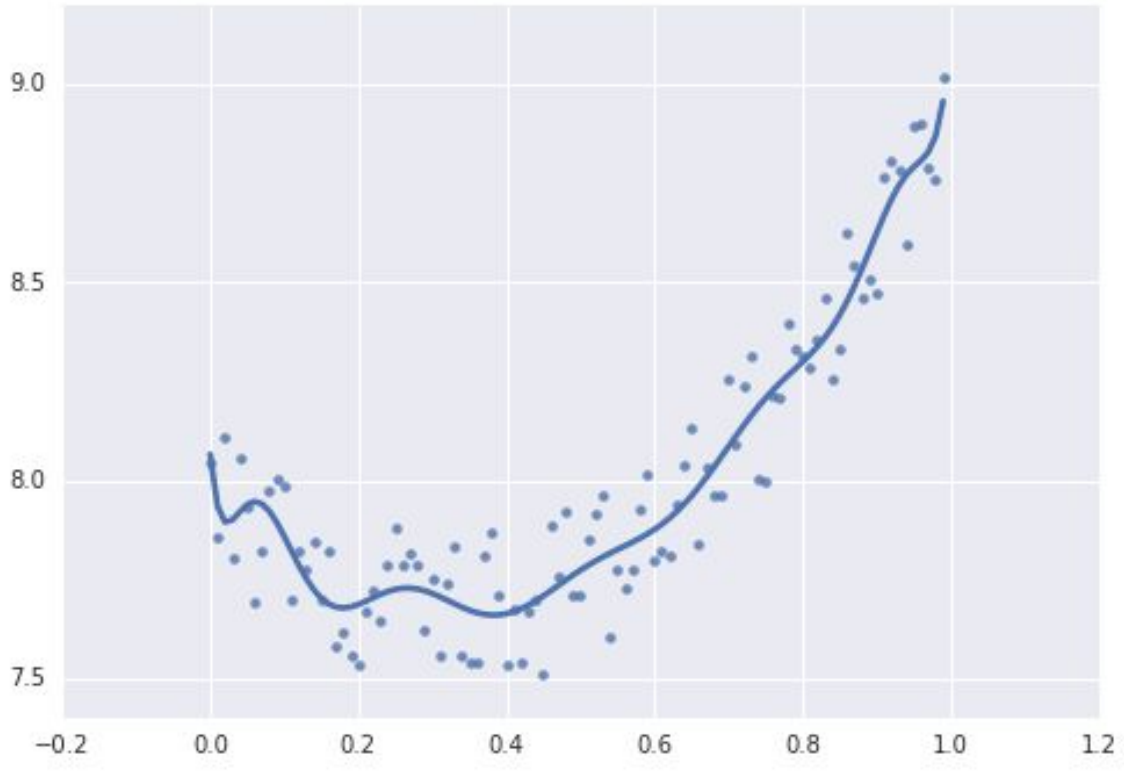
Fit a Curve

- A curve would give us less error



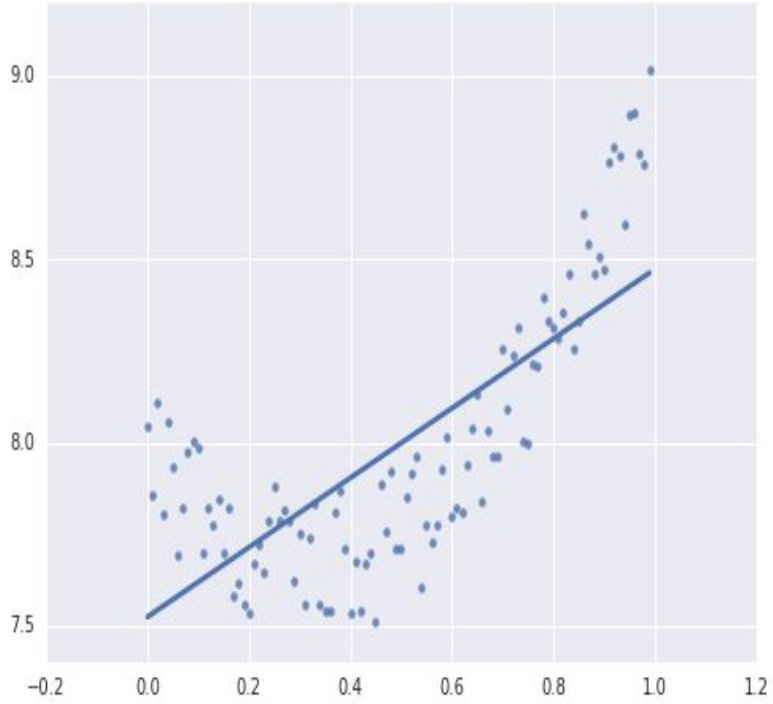
Over-Fitting

- A complex curve may result in less error on the training data, but may not make accurate predictions on future data

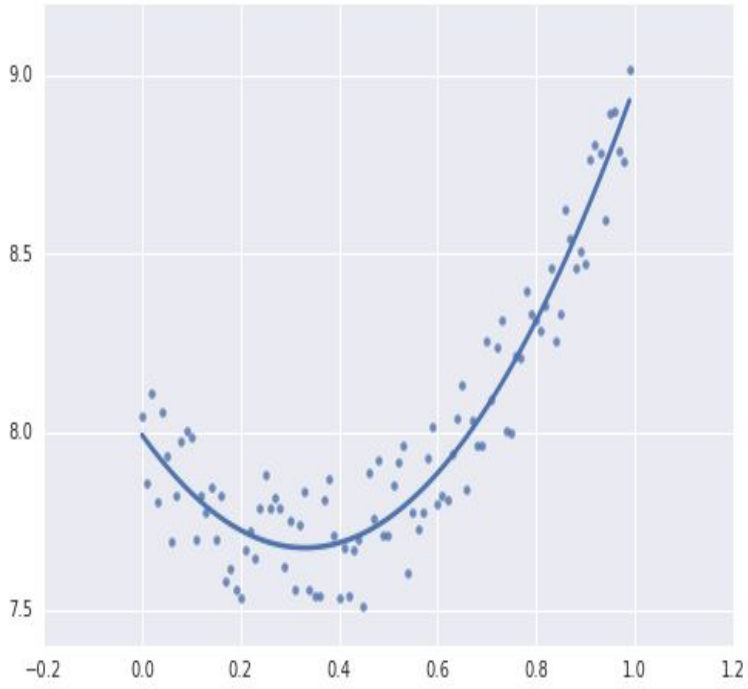


Optimization Is Iterative

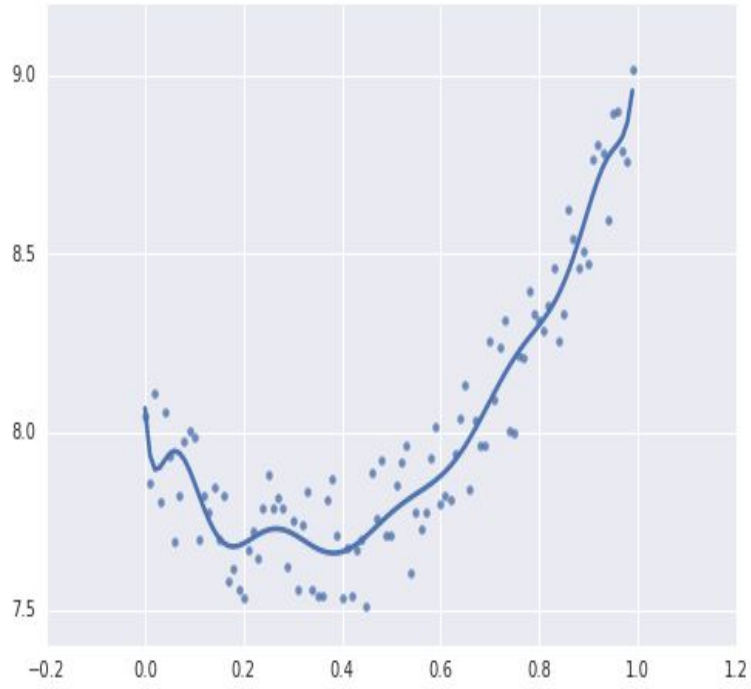
Underfit



Fit



Overfit



Chapter Concepts

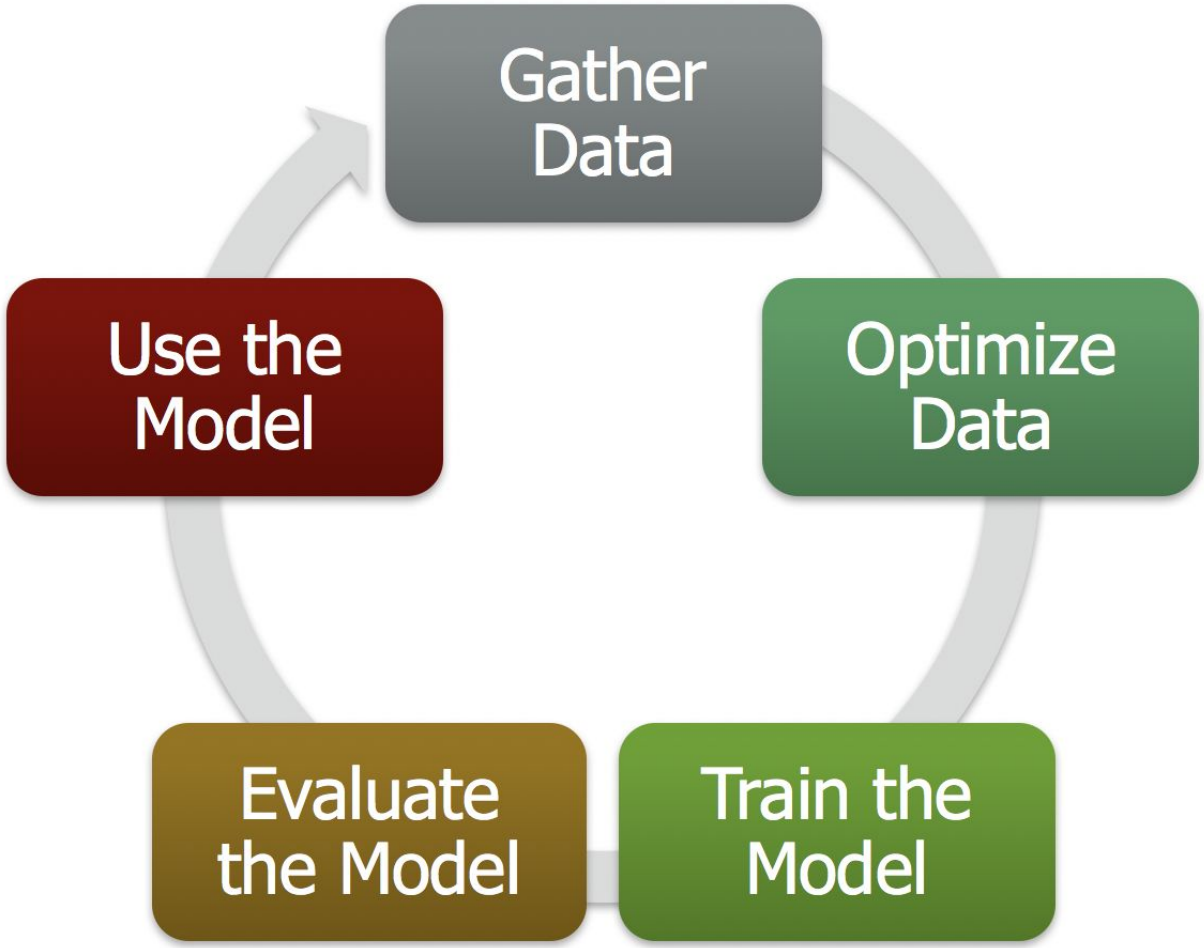
Machine Learning Basics

Building Machine Learning Models

Leveraging Existing Machine Learning Models

Chapter Summary

Machine Learning Process



Machine Learning Data

- *Features* are the data elements used to train the model
  - If predicting the sale price of a house, what might be some useful data elements that could be used for features?
- Training data must have a *target*, the historically correct answer and the value that the model is going to predict
  - E.g., the sale price of the house
- Data must be split into training and test sets
  - Test set is used after the model is trained to determine its accuracy
  - Typically 80% training and 20% testing

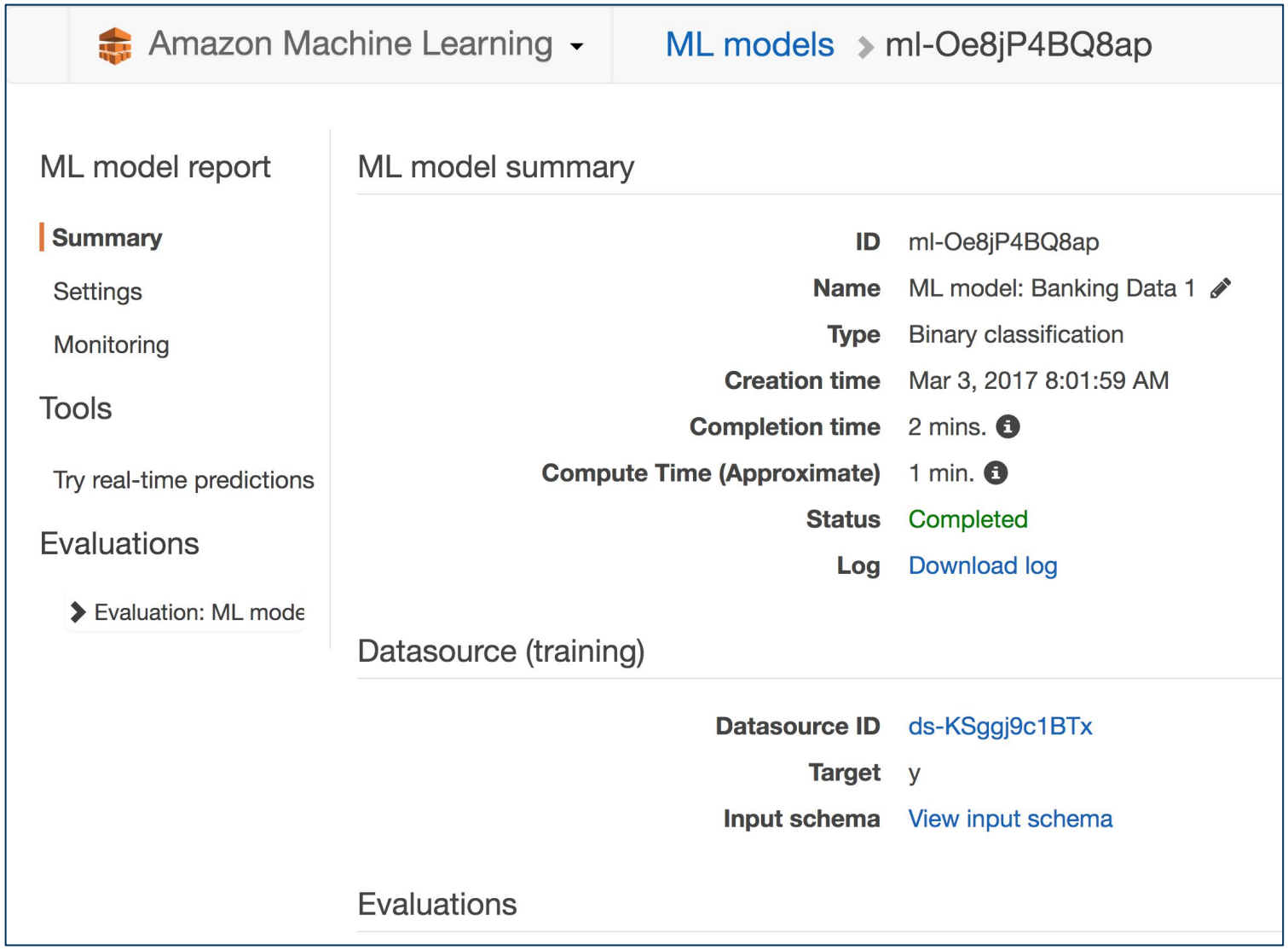
Feature Tuning

- Data needs to be optimized for machine learning
- Remove features that are not relevant
- Remove unique identifiers (IDs and timestamps)
  - May cause the model to “memorize” the training data
- Binary choices should be converted to 0’s and 1’s (true/false, yes/no, etc.)
- Fix inconsistencies
  - NYC, new_york, New York, New York City should be made the same
- Remove missing data or replace nulls with sensible default values
- Binning is used to group data items into relevant buckets
  - Instead of sale date, group sales by season
- Etc.

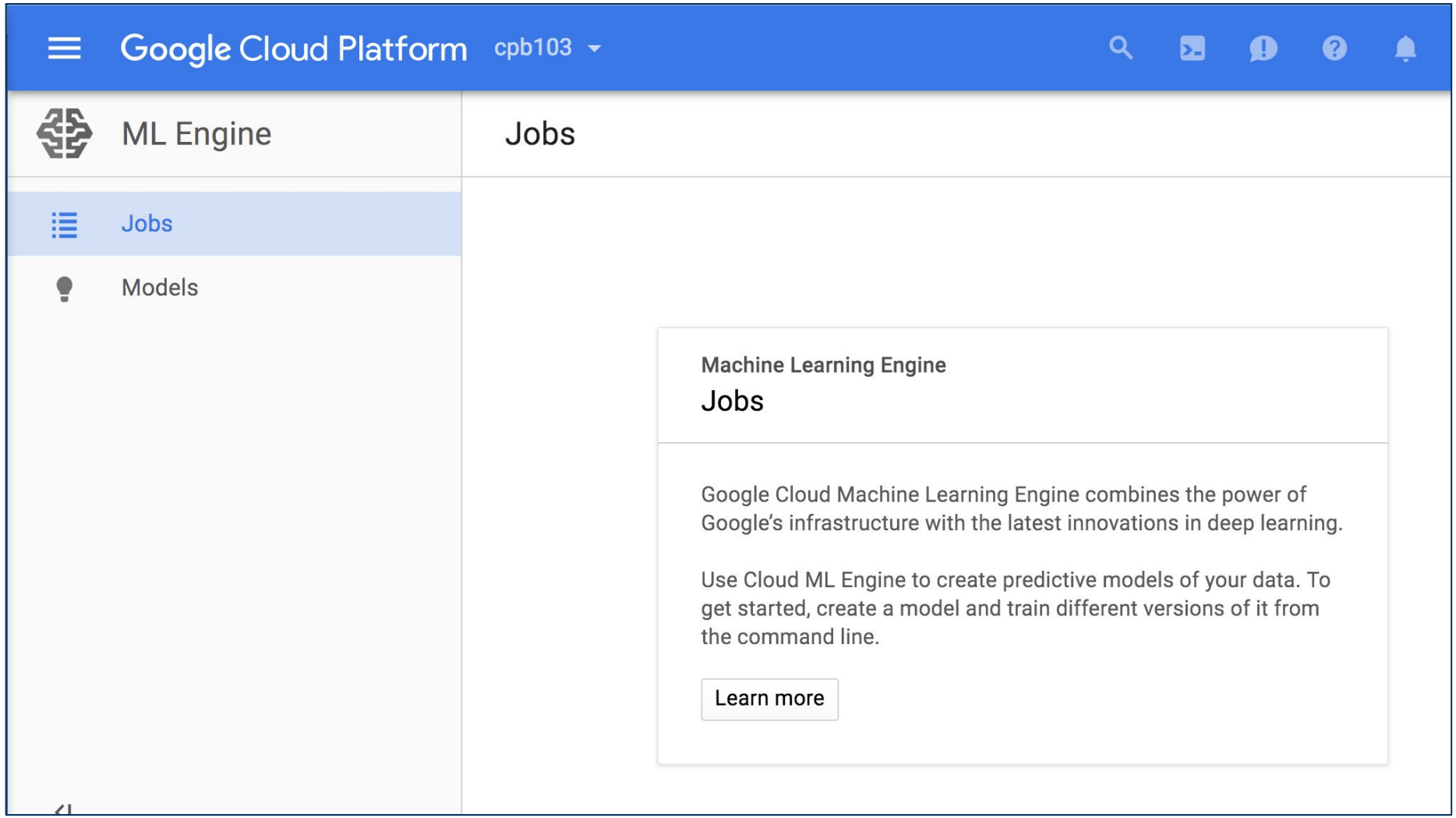
Training Models

- Real-world machine learning requires huge data sets and massive infrastructure for effective training
- Both AWS and Google Cloud Platform provide machine learning services
  - Do much of the heavy lifting while training
- Cloud infrastructure supports huge amounts of data

Amazon Machine Learning



Google Cloud ML



Exercise

- AWS Machine Learning
  https://storage.googleapis.com/roi-code-labs/codelabs/labs/793-aws-machine-learning/index.html
- GCP Machine Learning
  https://storage.googleapis.com/roi-code-labs/codelabs/labs/793-gcp-machine-learning/index.html

Chapter Concepts

Machine Learning Basics

Building Machine Learning Models

Leveraging Existing Machine Learning Models

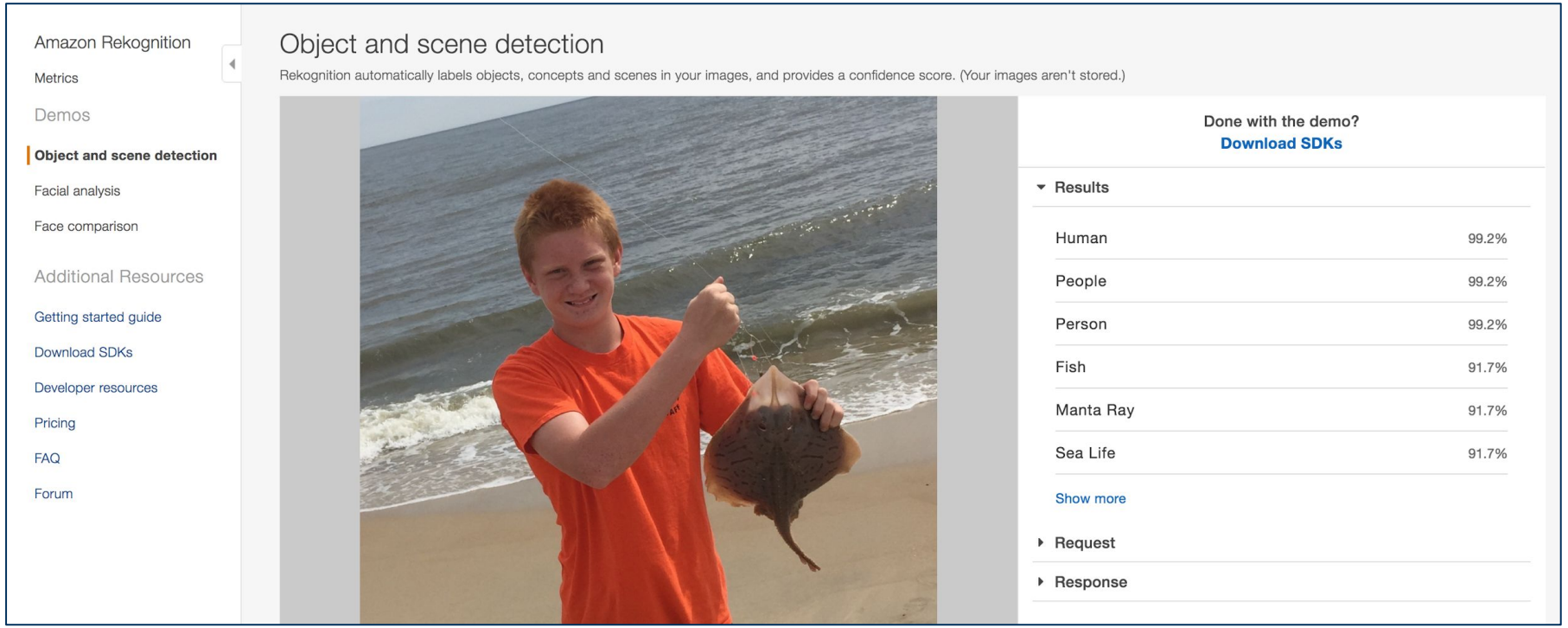
Chapter Summary

Utilizing Pre-Built Models

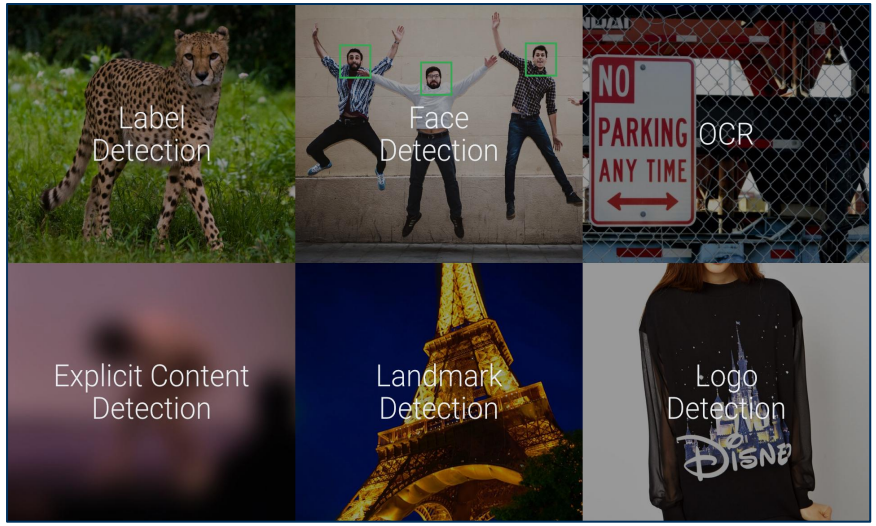
- Many models would be too difficult for most companies to create
  - Vision
  - Speech
  - Translation
  - Etc.
- Many complex machine learning models already exist and are provided by cloud vendors
- Easy to make use of in your own applications
  - Simple REST APIs are used to invoke the model

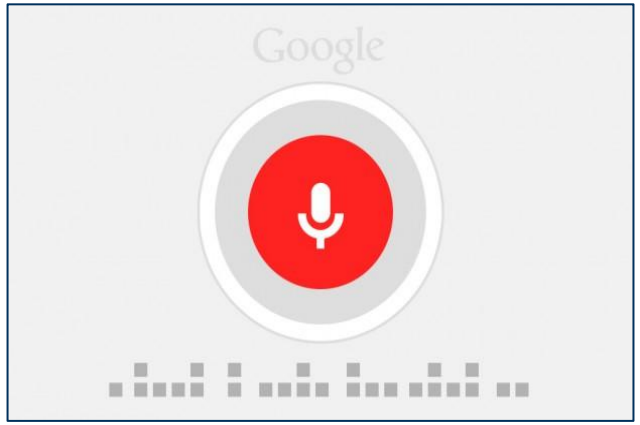
AWS Machine Learning APIs

- Lex for speech recognition and natural language understanding
- Polly for text to speech
- Rekognition for image recognition



Google Machine Learning APIs





- Translate API
- Vision API
- Speech API
- Language API

Exercise

- AWS Pre-Built Machine Learning Models
  https://storage.googleapis.com/roi-code-labs/codelabs/labs/793-aws-machine-learning-pre-built-models/index.html
- GCP Pre-Built Machine Learning Models
  https://storage.googleapis.com/roi-code-labs/codelabs/labs/793-gcp-machine-learning-pre-built-models/index.html

Chapter Concepts

Machine Learning Basics

Building Machine Learning Models

Leveraging Existing Machine Learning Models

Chapter Summary

Chapter Summary

- Machine learning is a way of programming based on data and math rather than traditional algorithms
- AWS and GCP provide services that make machine learning more accessible to traditional programmers
- There are advanced machine learning APIs provided by Google and Amazon that can be incorporated into your applications today

ROI TRAINING
MAXIMIZE YOUR TRAINING INVESTMENT

Programming Microservice and Big Data Applications in the Cloud

CHAPTER 7: COURSE SUMMARY

Course Summary

In this course, we have learned how to:

- Use automation for continuous integration and delivery
- Automate testing
- Design and program microservices using a REST architecture
- Integrate cloud APIs into microservice applications
- Leverage the cloud for Big Data processing
- Enable real-time data analysis
- Add machine learning capabilities to applications

ROI's Training Curricula

- Agile Development
- Big Data & Data Analytics
- Business Analysis
- Cloud Computing & Virtualization
- Google Cloud Platform
- ITIL & IT Service Management
- Java 8 / 7
- Leadership & Management Skills
- Microsoft Exchange 2013 / 2010
- .NET & Visual Studio
- Networking & IPv6
- New Hire Programs
- Oracle Database 12*c* / 11*g*
- OpenStack & Docker
- Project Management
- Python & Perl Programming
- Security
- SharePoint
- Software Analysis & Design
- Software Engineering
- SQL Server 2014 / 2012 / 2008 R2
- UNIX and Linux
- Web & Mobile Apps
- Windows 10 / 8.1 / 7
- Windows Server 2012 R2 / 2012 / 2008

*Please visit our website (www.ROItraining.com) for a complete list of offerings.

Ways to Stay in Touch

ROITRAINING
MAXIMIZE YOUR TRAINING INVESTMENT

ROITraining.com

Follow us!
@ROITraining

Connect with us!
/company/roi-training

Like us!
/ROITraining

The management and staff of **ROI Training** would like to thank you for your commitment to continuing education.

Our mission is to deliver customized technology and management training solutions to large corporations and government agencies around the world. We strive to provide business professionals with the skills and knowledge necessary to increase work performance. We hope the training facilitated by your ROI instructor enables you to successfully apply what you have learned to your work.

We wish you continued success in your career and hope to see you again in the near future.

Best regards,

Brian Reimer and *David Carey*
Co-Founders